Using Network Plug and Play

This chapter contains the following sections:

About Network Plug and Play

Network Plug and Play (PnP) is a software application that runs on a Cisco Nexus 9500 Series Switch (specifically, N9K-C9504, N9K-C9508, and N9K-C9516). The PnP feature provides a simple, secure, unified, and integrated offering to make a new branch or campus rollouts much easier, or for provisioning updates to an existing or a new network. This feature provides a unified approach to provision networks comprising multiple devices with a near-zero-touch deployment experience.

Simplified deployment reduces the cost and complexity and increases the speed and security of the deployments. The PnP feature helps simplify the deployment of any Cisco device by automating the following deployment-related operational tasks:

- Establishing initial network connectivity for a device.
- Delivering device configuration to the controller.
- Delivering software and firmware images to the controller.
- Provisioning local credentials of a switch.
- Notifying other management systems about deployment-related events.

The PnP is a client-server based model. The client (agent) runs on a Cisco Nexus 9500 Series Switch and the server (controller) runs on the Cisco DNA Controller.

PnP uses a secure connection to communicate between the agent and the controller. This communication is encrypted.

For information on configuring and managing the needed security certificate(s) for PnP functionality, see the Cisco Digital Network Architecture Center Security Best Practices Guide.

The PnP agent converge solutions that exist in a network into a unified agent and adds additional functionality to enhance the current solutions. The main objectives of the PnP agent is to provide consistent Day 0 deployment solution for all the deployment scenarios.

Features Provided by the Network Plug and Play (PnP) Agent

Day 0 Provisioning

Day 0 bootstrapping includes the configuration, image, and other files. When a device is powered on for the first time, the PnP discovery process, which is embedded in the device, gets enabled in the absence of a startup configuration file and attempts to discover the address of the PnP controller or server. The PnP agent uses methods such as DHCP, Domain Name System (DNS), and others to acquire the desired IP address of the PnP server.

When the PnP agent successfully acquires the IP address, it initiates a long-term, bidirectional Layer 3 connection with the server and waits for a message from the server. The PnP server application sends messages to the corresponding agent, requesting for information about the devices and the services to be performed on the device.

The agent running on the Cisco Nexus 9500 Series switch then configures the IP address on receiving the DHCP acknowledgment and establishes a secure channel with the controller to provision the configurations. The switch then upgrades the image and applies the configurations.

Discovery Methods

A PnP agent discovers the PnP controller or server using one of the following methods:

- DHCP-based discovery
- DNS-based discovery
- PnP connect

After the discovery, the PnP agent writes the discovered information into a file, which is then used to handshake with the PnP server (DNA controller/DNA-C).

The following tasks are carried out by the agent in the PnP discovery phase:

- Brings up all the interfaces.
- Sends a DHCP request in parallel for all the interfaces.
- On receiving a DHCP reply, configures the IP address and mask, default route, DNS server, domain name, and writes the PnP server IP in a lease-parsing file. Note that there is no DHCP client in Cisco Nexus Switches and static configuration is required.
- Brings down all the interfaces.

Note POAP is the first order of choice for Day 0 provisioning. Only when there is no valid POAP offer, PnP discovery is attempted. Also, PnP is supported only on Cisco Nexus 9000 EoR models N9K-C9504, N9K-C9508, and N9K-C9516. PnP is not supported on Cisco Nexus 9000 ToRs.

DHCP-Based Discovery

When the switch is powered on and if there is no startup configuration, the PnP starts with DHCP discovery. DHCP discovery obtains the PnP server connectivity details.

The PnP agent configures the following:

- IP address
- Netmask
- Default gateway
- DNS server

- Domain name

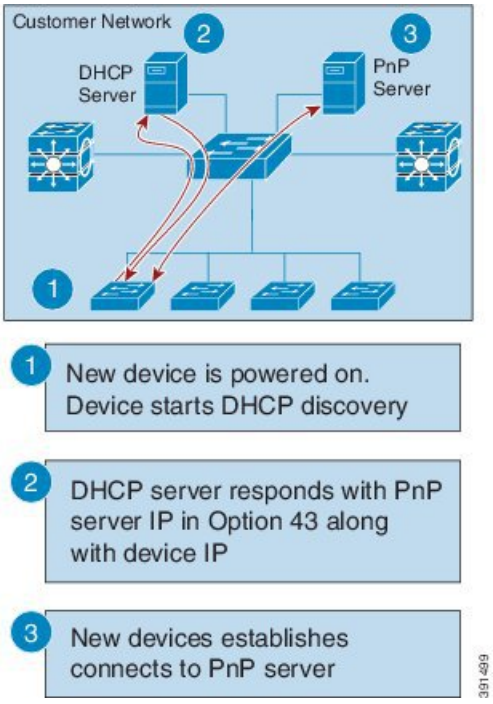
If the agent configuration fails, you should manually intervene and configure the switch.

DHCP discovery has the following flow:

- Power on the switch.
- Switch will boot up, the PnP process will be started, as there is no configuration present.
- Start DHCP discovery.
- DHCP Server replies with the PnP server configuration.
- PnP agent handshakes with the PnP server.
- Download the image, install, and reload.
- Download and apply the configuration from the controller.

A device with no startup configuration in the NV-RAM triggers the day 0 provisioning and goes through the POAP process (as detailed in m_using_poweron_auto_provisioning_92x.ditamap#id_70221). When there is no valid POAP offer, the PnP agent is initiated. The DHCP server can be configured to insert additional information using vendor-specific Option 43. Upon receiving Option 60 from the device with the string (cisco pnp), to pass on the IP address or hostname of the PnP server to the requesting device. When the DHCP response is received by the device, the PnP agent extracts the Option 43 from the response to get the IP address or the hostname of the PnP server. The PnP agent then uses this IP address or hostname to communicate with the PnP server.

Figure 1: DHCP Discovery Process for PnP Server



DNS-Based Discovery

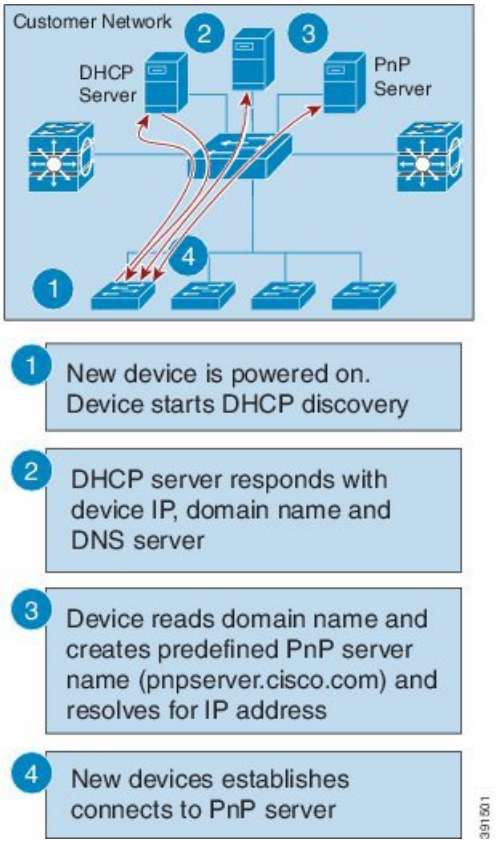
When the DHCP discovery fails to get the PnP server, the agent falls back to DNS-based discovery. To start the DNS-based discovery, the following information is required from DHCP:

- IP address and netmask
- Default gateway

- DNS server IP
- Domain name

The agent obtains the domain name of the customer network from the DHCP response and constructs the fully qualified domain name (FQDN). The following FQDN is constructed by the PnP agent using a preset deployment server name and the domain name information for the DHCP response. The agent then looks up the local name server and tries to resolve the IP address for the above FQDN.

Figure 2: DNS Lookup for pnpserver.[domainname].com



Note The device reads domain name and creates predefined PnP server name as pnpserver.[domain name].com, for example; pnpserver.cisco.com.

Plug and Play Connect

When the DHCP and the DNS discovery fail, the PnP agent discovers and communicates with Cisco Cloud-based deployment service for initial deployment. The PnP agent directly opens an HTTPS channel using the Python library, which internally invokes OpenSSL to talk with cloud for configuration.

Cisco Power On Auto Provisioning

Cisco Power On Auto Provisioning (PoAP) communicates with the DHCP and TFTP servers to download the image and configurations. With the introduction of the PnP feature, PnP and PoAP coexist together in a Cisco Nexus 9500 Series Switch. PoAP and PnP interworking has the following processes:

- PoAP starts first when no start-up configuration is present in the system.
- PnP starts later if PoAP does not get provisioned.
- PoAP and PnP discover the controller alternatively.
- The controller discovery process continues until a controller or until the admin aborts auto provision.
- The process (POAP or PnP) that finds the controller continues provisioning and the other process that does not find the controller is notified and eventually terminated.

Services and Capabilities of the Network Plug and Play Agent

The PnP agent performs the following tasks:

- Backoff
- Capability
- CLI execution
- Configuration upgrade
- Device information
- Certificate install
- Image install
- Redirection

Note The PnP controller or server provides an optional checksum tag to be used in the image installation and configuration upgrade service requests by the PnP agent. When the checksum is provided in a request, the image install process compares the checksum against the current running image checksum.

If the checksums are same, the image being installed or upgraded is the same as the current image running on the device. The image install process will not perform any other operation in this scenario.

If the checksums are not the same, the new image will be copied to the local file system, and the checksum will be calculated again and compared with the checksum provided in the request. If they are the same, the image install process continues to install the new image or upgrade the device to the new image. If the checksums are not the same, the process exits with an error.

Backoff

A Cisco NX-OS device that supports PnP protocol, which uses HTTP transport, requires the PnP agent to send the work request to the PnP server continuously. If the PnP server does not have any scheduled or outstanding PnP service for the PnP agent to execute, the continuous no-operation work requests exhaust both the network bandwidth and the device resources. This PnP backoff service allows the PnP server to inform the PnP agent to rest for the specified time and call back later.

Capability

Capability service request is sent by the PnP server to the PnP agent on a device to query the supported services by the agent. The server then sends an inventory service request to query the device's inventory information; and then sends an image installation request to download an image and install it. After getting the response from the agent, the list of supported PnP services and features are enlisted and returned back to the Server.

CLI Execution

Cisco NX-OS supports two modes of command execution, privileged EXEC mode and global configuration mode. Most of the EXEC commands are one-time commands, such as **show** commands, which show the current configuration status, and clear commands, which clear counters or interfaces. The EXEC commands

are not saved when a device reboots. Configuration mode commands allow user to make changes to the running configuration. If you save the configuration, these commands are saved when a device reboots.

Configuration Upgrade

Two types of configuration upgrades takes place in a Cisco device—copying new configuration files to the startup configuration and copying new configuration files to the running configuration.

Copying new configuration files to the startup configuration—A new configuration file is copied from the file server to the device using the **copy** command, and the file check task is performed to check the validity of the file. If the file is valid, the file is copied to the startup configuration. The previous configuration file is backed up if enough disk space is available. The new configuration comes into effect when the device reloads again.

Copying new configuration files to the running configuration—A new configuration file is copied from the file server to the device using the **copy** command or **configure replace** command. Replace and rollback of configuration files may leave the system in an unstable state if rollback is performed inefficiently. Therefore, configuration upgrade by copying the files is preferred.

Device Information

The PnP agent provides the capability to extract device inventory and other important information to the PnP server on request. The following device-profile request types are supported:

- all—Returns complete inventory information, which includes unique device identifier (UDI), image, hardware, and file system inventory data.
- filesystem—Returns file system inventory information, which includes file system name and type, local size (in bytes), free size (in bytes), read flag, and write flag.
- hardware—Returns hardware inventory information, which includes hostname, vendor string, platform name, processor type, hardware revision, main memory size, I/O memory size, board ID, board rework ID, processor revision, mid-plane revision, and location.
- image—Returns image inventory information, which includes version string, image name, boot variable, return to ROMMON reason, bootloader variable, configuration register, configuration register on next boot, and configuration variables.
- UDI—Returns the device UDI.

Certificate Install

Certificate install is a security service through which a PnP server requests the PnP agent on a device for trust pool or trust point certificate installation or uninstallation. This service also specifies the agent about the primary and backup servers for reconnection. The following prerequisites are required for a successful certificate installation:

- The server from which the certificate or trust pool bundle needs to be downloaded should be reachable.
- There should not be any permission issues to download the certificate or the bundle.
- The PKI API should be available and accessible for the PnP agent so that the agent could call to download and install the certificate or the bundle.
- There is enough memory on the device to save the downloaded certificate or bundle.

PnP Agent

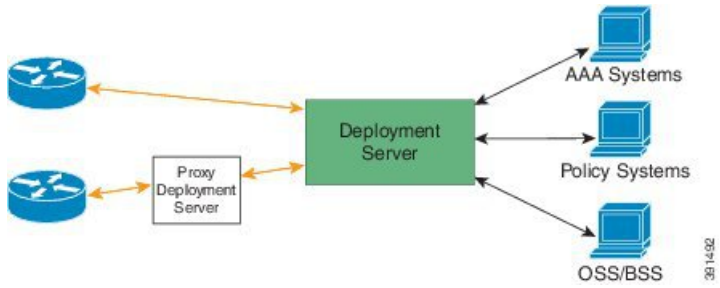
The PnP agent is an embedded software component that is present in all Cisco network devices that support simplified deployment architecture. The PnP agent understands and interacts only with a PnP server. The PnP agent first tries to discover a PnP server, with which it can communicate. After a server is found and connection established, the agent performs deployment-related activities such as configuration, image and file updates

by communicating with the server. It also notifies the server of all interesting deployment-related events such as out-of-band configuration changes and new device connections on an interface.

PnP Server

The PnP server is a central server that encodes the logic of managing and distributing deployment information (images, configurations, files, and licenses) for the devices being deployed. The server communicates with the agent on the device that supports the simplified deployment process using a specific deployment protocol.

Figure 3: Simplified Deployment Server



The PnP server also communicates with proxy servers such as deployment applications on smart phones and PCs, or other PnP agents acting as Neighbor Assisted Provisioning Protocol (NAPP) servers, and other types of proxy deployment servers such as VPN gateways.

The PnP server can redirect the PnP agent to another deployment server. A common example of redirection is a PnP server redirecting a device to communicate with it directly after sending the bootstrap configuration through a NAPP server. A PnP server can be hosted by an enterprise. This solution allows for a cloud-based deployment service provided by Cisco. In this case, a device discovers and communicates with Cisco cloud-based deployment service for initial deployment. After that, it can be redirected to the customer's deployment server.

In addition to communicating with the devices, the server interfaces with a variety of external systems such as authentication, authorizing, and accounting (AAA) systems, provisioning systems, and other management applications.

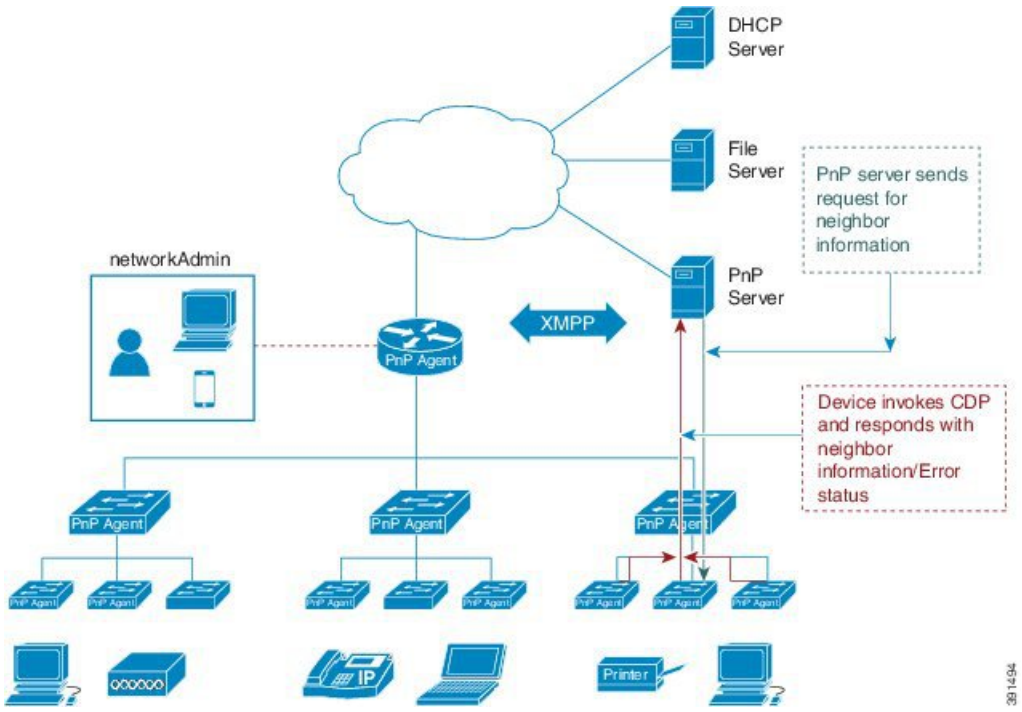
PnP Agent Deployment

The following steps indicate the PnP agent deployment procedure on Cisco devices:

1. A Cisco device with a PnP agent contacts the PnP server, requesting for a task, that is, the PnP agent sends UDI along with a request for work.
2. If the PnP server has a task for the device, for example, image installation, configuration, upgrade, and so on, it sends a work request.
3. After the PnP agent receives the work request, it executes the task and sends back a reply to the PnP server about the task status, that is whether it is successful or if an error has occurred, and the corresponding information that is requested.

PnP Agent Network Topology

Figure 4: Network Topology of Cisco PnP Agent Deployment



PnP Agent Initialization

The PnP agent is enabled by default, but can be initiated on a device when the startup configuration is not available.

Absence of Startup Configuration

New Cisco devices are shipped to customers with no startup configuration file in the NVRAM of the devices. When a new device is connected to a network and powered on, the absence of a startup configuration file on the device automatically triggers the PnP agent to discover the PnP server IP address.

CLI Configuration for the PnP Agent

PnP supports devices that are using VLAN 1 by default.

Guidelines and Limitations for Network Plug and Play

Network Plug and Play (PnP) guidelines and limitations are as follows:

- Beginning with NX-OS 9.2(3), PnP is supported on the management port of Cisco Nexus 9500 platform switches.
- PnP runs on both the in-band and the management interfaces. In-band is supported only on FX-series line cards (specifically N9K-X9736C-FX for PnP).
- The PnP deployment method depends on the discovery process that is required for finding the PnP controller or server.
- The discovery mechanism must be deployed, either as a DHCP server discovery process or a Domain Name Server (DNS) discovery process, before launching PnP.

- Configure the DHCP server or the DNS server before deploying PnP.
- The PnP server must communicate with the PnP agent.
- PnP connect does not require a DHCP or DNS configuration.
- IPv6 support for PnP is not available for Cisco Nexus 9500 Series devices.

Cisco DNA Center Support

The following guidelines and limitations are specific for PnP connectivity to the Cisco DNA Center:

- Cisco DNA Center supports the following functionality on the Cisco Nexus 9504, Cisco Nexus 9508, and Cisco Nexus 9516 switches:
  - Discovery
  - Inventory
  - Topology
  - Template Programmer
  - Software Image Management
  - Basic Monitoring
- The following PnP guidelines and limitations are only for the Cisco DNA Center version 1.2.6 and earlier:
  - The startup configuration that is provided during plug and play must ensure that the connectivity for the interface that is connected to the Cisco DNA Center remains intact.
  - The system image .bin and startup configuration must be uploaded to the Cisco DNA Center.
  - The bootflash must have enough space to download the image and configurations from the Cisco DNA Center.

    Click here for the user documentation for the Cisco DNA Center.

Troubleshooting Examples for Network Plug and Play

Example: Troubleshooting PnP

The following examples shows the PnP troubleshooting command outputs:

```
Switch# show pnp status
PnP Agent is running
server-connection
    status: Success
    time: 08:41:26 Jan 11
interface-info
    status: Success
    time: 08:34:00 Jan 11
device-info
    status: Success
    time: 08:33:46 Jan 11
config-upgrade
    status: Success
```

```
    time: 08:31:36 Jan 11
capability
    status: Success
    time: 08:33:50 Jan 11
backoff
    status: Success
    time: 08:41:26 Jan 11
topology
    status: Success
    time: 08:33:54 Jan 11


Switch# show pnp version
PnP Agent Version Summary

PnP Agent: 1.6.0
Platform Name: nxos
PnP Platform: 1.5.0.rc2


Switch# show pnp profiles
Created by              UDI
DHCP Discovery  PID:N9K-C9504,VID:V01,SN:FOX1813GCZ8

     Primary transport: https
     Address: 10.105.194.248
     Port: 443
     CA file: /etc/pnp/certs/trustpoint/pnplabel

     Work-Request Tracking:
         Pending-WR: Correlator=
Cisco-PnP-POSIX-nxos-1.6.0-21-589a466a-0d88-427b-a17e-69afb7d0a226-1
         Last-WR:    Correlator=
Cisco-PnP-POSIX-nxos-1.6.0-20-ab225de4-b0ef-46c5-9c4f-e3bd9f7c6b87-1
     PnP Response Tracking:
         Last-PR:    Correlator=
Cisco-PnP-POSIX-nxos-1.6.0-20-ab225de4-b0ef-46c5-9c4f-e3bd9f7c6b87-1


Switch# show pnp lease
{
    "lease": {
         "uptime": "Fri Jan 11 05:32:17 2019",
        "intf": "Vlan1",
         "ip_addr": "10.77.143.239",
         "mask": "255.255.255.0",
         "gw": "10.77.143.1",
         "domain": "",
         "opt_43": "5A1D;B2;K4;I10.105.194.248;J80",
         "lease": "3600",
         "server": "10.77.143.231",
         "vrf": "1"
     }
}


Switch# show pnp internal trace

1) Event:E_DEBUG, length:49, at 907122 usecs after Fri Jan 11 08:30:44 2019
    [104] pnp_ascii_gen: ascii gen completed rcode[0]

2) Event:E_DEBUG, length:16, at 907094 usecs after Fri Jan 11 08:30:44 2019
    [104] pss type: 5

3) Event:E_DEBUG, length:31, at 907069 usecs after Fri Jan 11 08:30:44 2019
    [104] Entering pnp_ascii_gen_cfg
```

```
4) Event:E_DEBUG, length:22, at 907061 usecs after Fri Jan 11 08:30:44 2019
    [104] Calling Ascii gen

5) Event:E_DEBUG, length:16, at 907051 usecs after Fri Jan 11 08:30:44 2019
    [104] pss type: 2

6) Event:E_DEBUG, length:49, at 907018 usecs after Fri Jan 11 08:30:44 2019
    [104] pnp_ascii_gen: fu_num_acfg_pss_entries[0x2]

7) Event:E_DEBUG, length:49, at 973813 usecs after Fri Jan 11 08:29:51 2019
    [104] pnp_ascii_gen: ascii gen completed rcode[0]

8) Event:E_DEBUG, length:16, at 973787 usecs after Fri Jan 11 08:29:51 2019
    [104] pss type: 5

9) Event:E_DEBUG, length:31, at 973760 usecs after Fri Jan 11 08:29:51 2019
    [104] Entering pnp_ascii_gen_cfg

10) Event:E_DEBUG, length:22, at 973751 usecs after Fri Jan 11 08:29:51 2019
    [104] Calling Ascii gen

11) Event:E_DEBUG, length:16, at 973742 usecs after Fri Jan 11 08:29:51 2019
    [104] pss type: 2

12) Event:E_DEBUG, length:49, at 973707 usecs after Fri Jan 11 08:29:51 2019
    [104] pnp_ascii_gen: fu_num_acfg_pss_entries[0x2]

13) Event:E_DEBUG, length:35, at 535794 usecs after Fri Jan 11 08:04:15 2019
    [104]  pnp_pi_spawn_finalize pid 690

14) Event:E_DEBUG, length:41, at 228291 usecs after Fri Jan 11 08:04:13 2019
    [104] + pnp_pi_spawn child_pid: 0xdd526da0

15) Event:E_DEBUG, length:76, at 132853 usecs after Fri Jan 11 08:03:26 2019
    [104] Rx: Direction: PnP PI -> PnP PD, Type: Device Provisioned, Cfg: Present

16) Event:E_DEBUG, length:35, at 440380 usecs after Fri Jan 11 08:03:18 2019
    [104] !!! ACKED Unconfigure Ret:1!!!

17) Event:E_DEBUG, length:61, at 440347 usecs after Fri Jan 11 08:03:18 2019
    [104] Tx: Direction: Max, Type: DHCP Unconfigure Done, Len: 16

18) Event:E_DEBUG, length:35, at 440331 usecs after Fri Jan 11 08:03:18 2019
    [102] Unknown timer cancel requested

19) Event:E_DEBUG, length:35, at 440311 usecs after Fri Jan 11 08:03:18 2019
    [104] pnp_pss_runtime_commit success

20) Event:E_DEBUG, length:57, at 440103 usecs after Fri Jan 11 08:03:18 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

21) Event:E_DEBUG, length:23, at 440051 usecs after Fri Jan 11 08:03:18 2019
    [104] - pnp_vsh_halt:206

22) Event:E_DEBUG, length:17, at 950291 usecs after Fri Jan 11 08:03:15 2019
    [104] Adding "end"

23) Event:E_DEBUG, length:58, at 950269 usecs after Fri Jan 11 08:03:15 2019
    [104] Adding "configure terminal ; no clock protocol none "

24) Event:E_DEBUG, length:33, at 945994 usecs after Fri Jan 11 08:03:15 2019
    [104] - pnp_vsh_config_l3_intf:788
```

```
25) Event:E_DEBUG, length:29, at 945979 usecs after Fri Jan 11 08:03:15 2019
    [104] + pnp_vsh_config_l3_intf

26) Event:E_DEBUG, length:39, at 945963 usecs after Fri Jan 11 08:03:15 2019
    [104] Adding "no feature interface-vlan"

27) Event:E_DEBUG, length:32, at 945932 usecs after Fri Jan 11 08:03:15 2019
    [104] Adding "configure terminal"

28) Event:E_DEBUG, length:40, at 945886 usecs after Fri Jan 11 08:03:15 2019
    [104] Got Semaphore, vsh halt continue...

29) Event:E_DEBUG, length:46, at 945870 usecs after Fri Jan 11 08:03:15 2019
    [104] sem_timedwait Success, Start VSH clean up

30) Event:E_DEBUG, length:19, at 945843 usecs after Fri Jan 11 08:03:15 2019
    [104] + pnp_vsh_halt

31) Event:E_DEBUG, length:35, at 945831 usecs after Fri Jan 11 08:03:15 2019
    [104] pnp_pss_runtime_commit success

32) Event:E_DEBUG, length:57, at 945643 usecs after Fri Jan 11 08:03:15 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

33) Event:E_DEBUG, length:33, at 945607 usecs after Fri Jan 11 08:03:15 2019
    [104] !!! Received Unconfigure !!!

34) Event:E_DEBUG, length:74, at 945578 usecs after Fri Jan 11 08:03:15 2019
    [104] Rx: Direction: PnP PI -> PnP PD, Type: DHCP Unconfigure, Cfg: Present

35) Event:E_DEBUG, length:49, at 789616 usecs after Fri Jan 11 08:01:52 2019
    [104] pnp_ascii_gen: ascii gen completed rcode[0]

36) Event:E_DEBUG, length:16, at 789579 usecs after Fri Jan 11 08:01:52 2019
    [104] pss type: 5

37) Event:E_DEBUG, length:31, at 789522 usecs after Fri Jan 11 08:01:52 2019
    [104] Entering pnp_ascii_gen_cfg

38) Event:E_DEBUG, length:22, at 789514 usecs after Fri Jan 11 08:01:52 2019
    [104] Calling Ascii gen

39) Event:E_DEBUG, length:16, at 789506 usecs after Fri Jan 11 08:01:52 2019
    [104] pss type: 2

40) Event:E_DEBUG, length:49, at 789489 usecs after Fri Jan 11 08:01:52 2019
    [104] pnp_ascii_gen: fu_num_acfg_pss_entries[0x2]

41) Event:E_DEBUG, length:35, at 789365 usecs after Fri Jan 11 08:01:52 2019
    [104] pnp_pss_runtime_commit success

42) Event:E_DEBUG, length:57, at 789135 usecs after Fri Jan 11 08:01:52 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

43) Event:E_DEBUG, length:26, at 789096 usecs after Fri Jan 11 08:01:52 2019
    [104] Phase Init -> Monitor

44) Event:E_DEBUG, length:35, at 788967 usecs after Fri Jan 11 08:01:52 2019
    [104]  pnp_pi_spawn_finalize pid 1c9

45) Event:E_DEBUG, length:41, at 831561 usecs after Fri Jan 11 08:01:49 2019
    [104] + pnp_pi_spawn child_pid: 0xffff7e28

46) Event:E_DEBUG, length:45, at 831550 usecs after Fri Jan 11 08:01:49 2019
```

```
    [104] Have startup config, Starting PnP PI....

47) Event:E_DEBUG, length:40, at 831538 usecs after Fri Jan 11 08:01:49 2019
    [104] Posix log directory creation failed

48) Event:E_DEBUG, length:50, at 831479 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_fire_event: PNP_EVENT_HAVE_STARTUP_CONFIG

49) Event:E_DEBUG, length:35, at 831465 usecs after Fri Jan 11 08:01:49 2019
    [104] Inside : pnp_other_msg_handler

50) Event:E_DEBUG, length:80, at 831437 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_get_data_from_queue: dequeued event 0x1102e0cc 25/cat 11 from pending Q

51) Event:E_DEBUG, length:50, at 831368 usecs after Fri Jan 11 08:01:49 2019
    [104] Injecting Event PNP_EVENT_HAVE_STARTUP_CONFIG

52) Event:E_DEBUG, length:59, at 831303 usecs after Fri Jan 11 08:01:49 2019
    [104] Have Startup Config, move the process state to monitor

53) Event:E_DEBUG, length:57, at 799379 usecs after Fri Jan 11 08:01:49 2019
    [104] Accelerating PnP, Break Point: Break Point PoAP Init

54) Event:E_DEBUG, length:35, at 799334 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit success

55) Event:E_DEBUG, length:57, at 799239 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

56) Event:E_DEBUG, length:23, at 799226 usecs after Fri Jan 11 08:01:49 2019
    [104] Phase None -> Init

57) Event:E_DEBUG, length:53, at 799200 usecs after Fri Jan 11 08:01:49 2019
    [104] Initilizing PnP-agent State machine curr_state 3

58) Event:E_DEBUG, length:35, at 799188 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit success

59) Event:E_DEBUG, length:57, at 799070 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

60) Event:E_DEBUG, length:26, at 798965 usecs after Fri Jan 11 08:01:49 2019
    [104] !!! Box is Online !!!

61) Event:E_DEBUG, length:35, at 798954 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit success

62) Event:E_DEBUG, length:57, at 798770 usecs after Fri Jan 11 08:01:49 2019
    [104] pnp_pss_runtime_commit: Stored values in runtime PSS

63) Event:E_DEBUG, length:70, at 370297 usecs after Fri Jan 11 07:55:41 2019
    [102] pnp_demux_mts(463): (Warning) unexpected mts msg (opcode - 7655)

64) Event:E_DEBUG, length:41, at 092701 usecs after Fri Jan 11 07:55:33 2019
    [104] PnP Init Internal subsystem, Done!!!

65) Event:E_DEBUG, length:32, at 089920 usecs after Fri Jan 11 07:55:33 2019
    [104] PnP Init Internal subsystem

Switch# show pnp posix_pi configs

/isan/etc/pnp/platform_config.cfg:

/isan/etc/pnp/file_paths.cfg:
```

```
/isan/etc/pnp/pnp_config.cfg:

/isan/etc/pnp/policy_discovery.conf:

/isan/etc/pnp/certs/platform.json:

/isan/etc/pnp/certs/pnp_status.json:

/isan/etc/pnp/certs/job_status.json:
```