Cisco Open Plug-n-Play Agent Configuration Guide, Cisco IOS XE Everest 16.4.1

Revised: December 6, 2016,

Feature Information for Open Plug-n-Play Agent

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for Open Plug-n-Play Agent

Feature Name	Releases	Feature Information
Open Plug-n-Play Agent	Cisco IOS XE Everest 16.4.1	The Cisco Open Plug-n-Play agent will not support the XMPP protocol from Cisco IOS XE Everest 16.4.1 release. The Cisco Open Plug-n-Play agent converges existing solutions into a unified agent and adds functionality to enhance the current deployment solutions. The following commands were introduced or modified by this feature: **backup device**, **backup profile**, **backup reconnect**, **backup transport http**, **backup transport https**, **backup transport xmpp socket**, **backup transport xmpp starttls**, **backup transport xmpp tls**, **device**, **pnp**, **profile**, **reconnect (pnp)**, **transport http**, **transport https**, **transport xmpp socket**, **transport xmpp starttls**, and **transport xmpp tls**.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Open Plug-n-Play Agent

- Cisco Open Plug-n-Play (PnP) deployment method depends on the type of discovery process as required by the customer.
- Deploy the discovery mechanism, either a DHCP server discovery process or a Domain Name Server (DNS) discovery process, before launching the PnP.
- Configure the DHCP server or the DNS server before deploying the PnP.
- Ensure that the PnP server talks to the PnP agent.
- The PnP agent enforces the PnP server to send user credentials for every request. Cisco recommends the usage of HTTP secure (HTTPS) protocol.

Note

- The terms Cisco Open Plug-n-Play, Plug-n-Play, PnP are interchangeably used in this guide and all mean the same.
- The terms PnP agent, agent, and deployment agent are interchangeably used in this guide and all mean the same.
- The terms PnP server, server, and deployment server are interchangeably used in this guide and all mean the same.

Restrictions for Open Plug-n-Play Agent

- Cisco Open Plug-n-Play (PnP) agent facilitates HTTP and HTTP secure (HTTPS) transport based communication with the server.
- HTTPS cannot be used on platforms where crypto-enabled images are not supported (also, do not use Secure Sockets Layer [SSL] or Transport Layer Security [TLS] protocols if crypto-enabled images are used).
- Non-VLAN 1 configuration-Cisco Network Plug and Play supports devices using VLAN 1 by default. To use a VLAN other than 1, adjacent upstream devices must use supported releases and configure the following global CLI command on the upstream device to push this CLI to the upcoming Plug and Play device: **pnp startup vlan x**. When you execute this command on an adjacent upstream device, the VLAN membership change does not happen on that device. However, all the active interfaces on the upcoming Plug and Play device are changed to the specified VLAN. This guideline applies to both routers and switches.

Note When you use the non-VLAN 1 feature, ensure that all the neighboring switch devices are running Cisco IOS XE Release 3.6.3 and not the 3.6.0, 3.6.1, or 3.6.2 releases. For more information about related caveat CSCut25533 that exists in these previous releases, see the Caveats section in the Release Notes for Cisco Network Plug and Play.

Information About Open Plug-n-Play Agent

Open Plug-n-Play Deployment Solution

The Cisco initiated Open Plug-n-Play (PnP) deployment solution supports the concept of redirection and includes a PnP agent, a PnP server, and other components. Simplified deployment process of any Cisco device automates the following deployment related operational tasks:

- Establishing initial network connectivity for the device
- Delivering device configuration
- Delivering software and firmware images
- Delivering licenses
- Delivering deployment script files
- Provisioning local credentials
- Notifying other management systems about deployment related events

Simplified deployment reduces the cost and complexity and increases the speed and security of deployments.

Cisco Open Plug-n-Play (PnP) agent is a software application that is running on a Cisco IOS or IOS-XE device. The PnP agent together with the PnP deployment server provides effortless deployment services. When a device is powered on for the first time, the PnP agent process wakes up in the absence of the startup config and attempts to discover the address of the PnP server. The PnP agent uses methods like 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 lived, bidirectional layer 3 connection with the server and waits for a message from the server. The PnP server application sends messages to the agent requesting for information and services to be performed on the device.

The PnP agent converges existing solutions into a unified agent and adds functionality to enhance the current solutions. The main objectives of PnP agent are:

- Provide consistent day 1 deployment solution for all the deployment scenarios.
- Add new features to improve existing solutions.
- Provide day 2 management framework mainly in the context of configuration and image upgrades.

Plug-n-Play Features

Some of the features that the Cisco Open Plug-n-Play (PnP) agent provides today are:

- Day 0 boot strapping—Configuration, image, licenses, and other files
- Day 2 management—Configuration and image upgrades and on-going monitoring of Simple Network Management Protocol (SNMP) and syslog messages.
- Open communication protocol—Enables customers and partners to write applications
- XML based payload over HTTP between the server and the agent.
- Security—Authentication and encrypted communication channel between the management app and the agent

- Deployment and management of devices behind firewall and Network Address Translation (NAT).
- Support for one-to-one and one-to-many communication
- Support for policy based deployment (product ID or location of the device)
- Deployment based on unique ID (Unique Device Identifier [UDI] or MAC)
- Unified solution across Cisco platforms (including IOS classic)
- Support for various deployment scenarios and use cases
- Zero-touch when possible, low-touch when needed

Plug-n-Play Agent Services and Capabilities

The services and capabilities of the Cisco Open Plug-n-Play (PnP) agent are as follows:

1. Backoff
2. CLI execution
3. Configuration upgrade
4. Device information
5. File transfer
6. Image install
7. License install
8. PnP tagging
9. Script execution
10. Topology information

Note The PnP server provides an optional checksum tag to be used in the image installation and config upgrade service requests by the PnP agent. When the checksum is provided in the 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 same, then the new image will be copied to the local file system, and checksum will again be calculated and compared against the checksum provided in the request. If same, the process will continue to install the new image or upgrade the device to new image. If now, the checksums are not same, the process will exit with error.

Backoff

A Cisco IOS device that supports PnP protocol (that uses HTTP transport), requires the PnP agent to send the work request to the PnP server continuously. In case the PnP server does not have any scheduled or outstanding PnP service for the PnP agent to execute, the continuous no operation work requests exhausts both network bandwidth and device resource. This PnP backoff service allows the PnP server to inform the PnP agent to rest for the specified time and call back later.

CLI Execution

Cisco IOS supports two modes of command execution—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 modes allow user to make changes to the running configuration. If you save the configuration, these commands are saved when a device reboots.

Note For **show** command request and response details and for all PnP configuration commands, see *Cisco Open Plug-n-Play Agent Command Reference*.

Configuration Upgrade

There are two types of configuration upgrades that can happen in a Cisco device—copying a new configuration files to startup configuration and copying new configuration files to running configuration.

Copying a new configuration files to startup configuration— The new configuration file is copied from the file server to the device through **copy** command and file check is performed to check the validity of the file. If the file is valid, then the file is copied to startup configuration. Backing up the previous configuration file will be done if there is enough disk space available. The new configuration is seen when the device reloads again.

Copying new configuration files to running configuration— The new configuration file is copied from the file server to the device through **copy** command or **configure replace** command. Configuration file replace and rollback may leave the system in an unstable state if rollback is performed efficiently. So configuration upgrade by copying the files is preferred.

Device Information

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

1. all—returns complete inventory information, which includes unique device identifier (UDI), image, hardware and file system inventory data.
2. 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.
3. 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, midplane revision, and location.
4. 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 variable.
5. UDI— returns device UDI.

File Transfer

The PnP file server hosts files that can be copied over by the deploying devices in the network. The file server can be a dedicated server hosting files or a part of the device hosting the PnP server. The PnP agent uses standard file transfer protocols to copy files from the remote file server to the device. If the device is running a crypto image then secured file transfer protocols such as SFTP, SCP, HTTPS are supported. For devices running non-crypto images, the PnP agent supports unsecured copy protocols such as FTP, TFTP, HTTP.

Image Install

Image installation service enables a PnP-enabled device to perform image upgrade on receiving a request from the PnP server.

An Image Install request can be classified for the following types of devices:

1. Standalone devices
2. High availability devices
3. Stackable devices
4. Cisco Catalyst 6000 devices

Standalone Devices
When the PnP agent on a standalone device receives a request from the PnP server, the agent parses the XML payload and identifies the request as an Image Upgrade request. The agent then creates an ImageInstall process, which identifies the request as a standalone image install request. The PnP agent populates the data structure defined by the ImageInstall service and passes it to the ImageInstall service.

The ImageInstall service then performs the following operations to successfully load the device with the new image:

1. Copies the image from the file server to a local disk (the file server information is provided by the PnP server in the request).
2. Configures the device to load the new image on next reload by executing the **boot system** command.
3. Reloads the device and sends a message to the PnP server.

High-Availability Devices
When the PnP agent is installed on a high-availability device, once ImageInstall service gets the data structure, the agent determines that the request is for a high-availability device. The active route processor (RP) which is running the PnP agent performs all the operations needed to install the image on both active and standby devices.

The ImageInstall service then performs the following operations to successfully load the devices with the new image:

1. Copies the image from the file server to a local disk of the device that is running the active RP. Information about the file server, image location, and destination is populated in the data structure.

   To reload the standby device with the same new image, the PnP agent running on the active device copies the image to the local disk on the standby device.

2. Configures the active device to load the new image on next reload by executing the **boot system** command.

   This causes the startup configuration to be synchronized with the standby device. Now, both active and standby devices have the same new image configured.

3. Sends a message to the PnP server that the active device is reloading the standby device.

   When the standby device comes up and reaches stateful switchover (SSO) state, the active device sends a message to the PnP server that it is going to reload itself, and then undergoes a reload.

   This causes a switchover to the standby device, which becomes the new active device.

Both active and standby devices are now running the new image.

Stackable devices

Cisco Catalyst 6000 devices

License Install

Cisco IOS automates installation of new licenses on PnP enabled Cisco devices. The PnP server sends work request to install new licenses on the Cisco device. When the PnP agent receives license install work request, it installs the new license on the device, configures the boot level, and reloads the device. The new license is applied when the device reloads. Once the service is completed, PnP agent sends success or failure work response to the PnP server.

PnP Tagging

Cisco IOS provides capability to assign tags to the devices for better grouping and tracking of all Cisco devices. The PnP agent provides XML service for configuring the tag information on the device and for propagating the tag information within the network using Cisco Discovery Protocol (CDP). The purpose of this service is for the PnP agents to get to know their tag information and to pass on this information to the PnP server upon request.

Script Execution

Cisco IOS provides capability to execute IOS shell scripts on all Cisco devices. Scripts can be executed by the PnP agent to perform post–install activities on the device. The script is copied from the file server by the agent and is executed on the running system. The post–install activities performed by these scripts consists of crypto key generation and other runtime configuration on the device.

Topology Information

By default, every Cisco device on the network runs Cisco Discovery Protocol (CDP). Through CDP, devices in the network discover their immediate neighbors and populate their databases with the attributes learnt or derived through the protocol. This neighbor information is stored in the database and is available on demand by the device to the PNP server. Typical neighbor information comprises neighboring device ID, software version, hardware platform, interface ip, and the port on which CDP messages are sent or received.

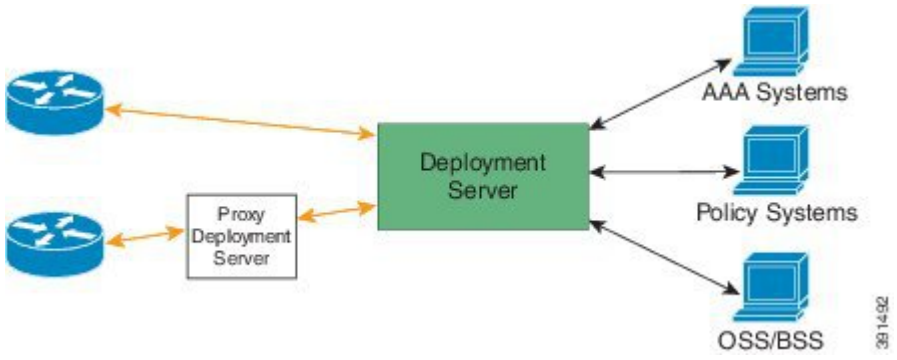
Plug-n-Play Agent

The Cisco Open Plug-n-Play (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. Once a server is found and connection established, the agent performs deployment related activities like configuration, image, license, and file updates by communicating with the server. It also notifies the server of all interesting deployment related events like out-of-band configuration changes and a new device connection on an interface.

Plug-n-Play Server

The Cisco Open Plug-n-Play (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 1: Simplified Deployment Server



The PnP server also communicates with proxy servers like 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 like VPN gateways.

The PnP server can redirect the 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's 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 like Authentication, Authorizing, and Accounting (AAA) systems, provisioning systems, and other management applications.

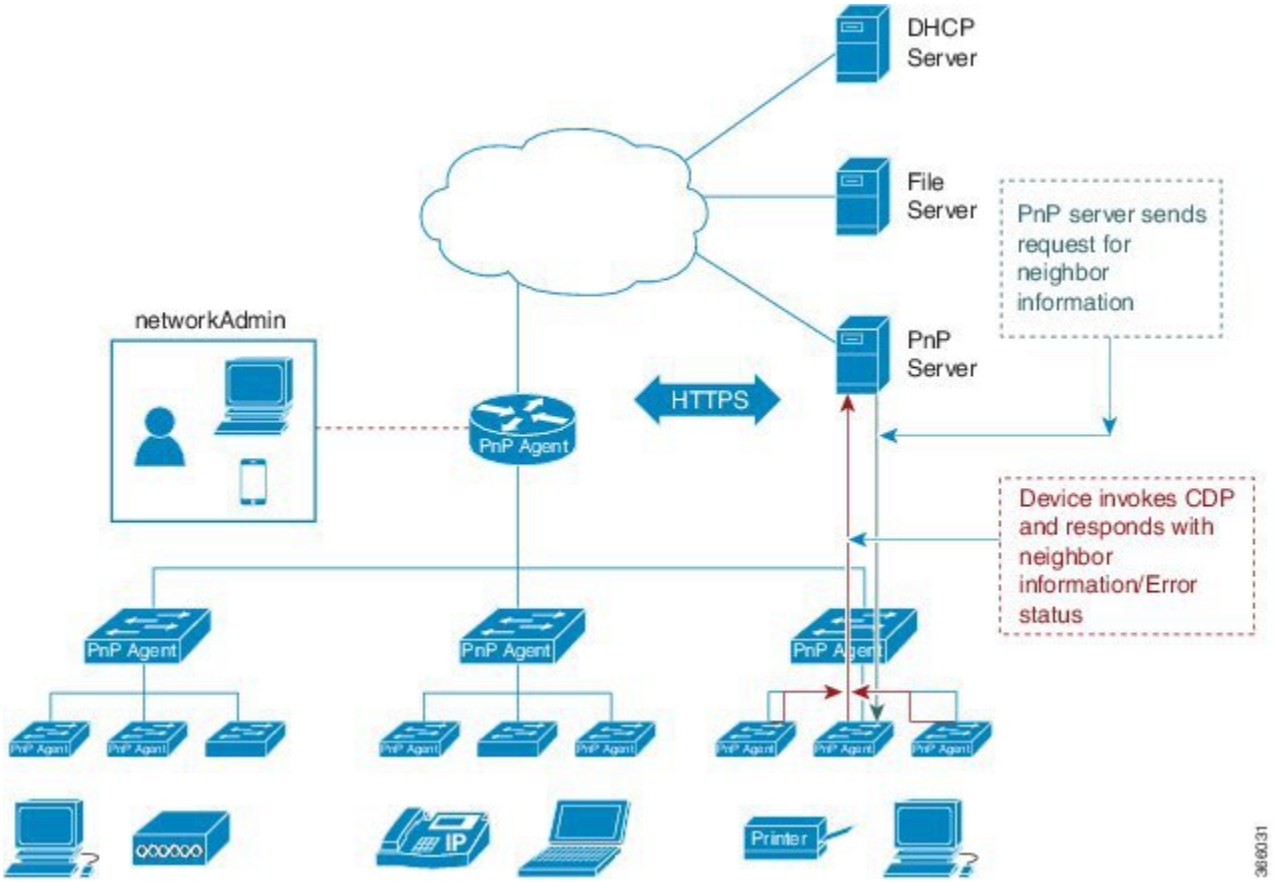
Plug-n-Play Agent Deployment

The following steps indicate the Cisco Open Plug-n-Play (PnP) agent deployment on Cisco devices:

1. The Cisco device, having PnP agent contacts the PnP server requesting for a task, that is, the PnP Agent sends its unique device identifier (UDI) along with a request for work.
2. The PnP server if it has any task for the device, sends a work request. For example, image install, config upgrade, and so on.
3. When the PnP agent receives the work request, executes the task and sends back a reply to the PnP server about the task status, whether it is a success or error, and the corresponding information requested.

Plug-n-Play Agent Network Topology

Figure 2: Network Topology of Cisco Open Plug-n-Play Agent Deployment



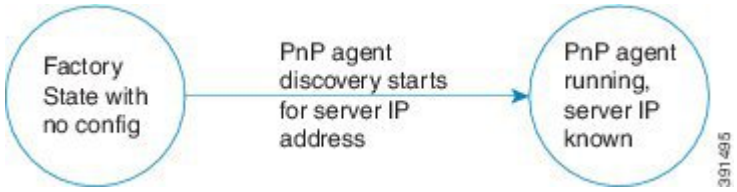
Plug-n-Play Agent Initialization

The Cisco Open Plug-n-Play (PnP) agent software is currently available on all Cisco IOS and IOS XE platforms, and is enabled by default. The PnP agent can be initiated on a device by the following ways:

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 will automatically trigger the Open Plug-n-Play (PnP) agent to discover the PnP server IP address.

Figure 3: State Diagram for PnP Trigger with no Startup Configuration



CLI Configuration for Open Plug-n-Play Agent
Network administrators may use CLI configuration mode to initiate the Open Plug-n-Play (PnP) agent process at any time. By configuring a PnP profile through CLI, a network administrator can start and stop the PnP agent on a device. When the PnP profile is configured with CLI, the device starts the PnP agent process which, in turn, initiates a connection with the PnP server using the IP address in the PnP profile.

Figure 4: State Diagram for PnP Trigger with CLI Configured PnP Profile

Open Plug-n-Play Agent Deployment Solutions

This section discusses the functionality of the Cisco Open Plug-n-Play (PnP) agent, exposed to the PnP server, for device deployment and management. The PnP agent deployment solution comprises the discovery process initiated by the agent, communication between the device, agent, and the server, and the PnP agent services. The PnP solution is described in detail in the following sections:

Plug-n-Play Agent Discovery Process

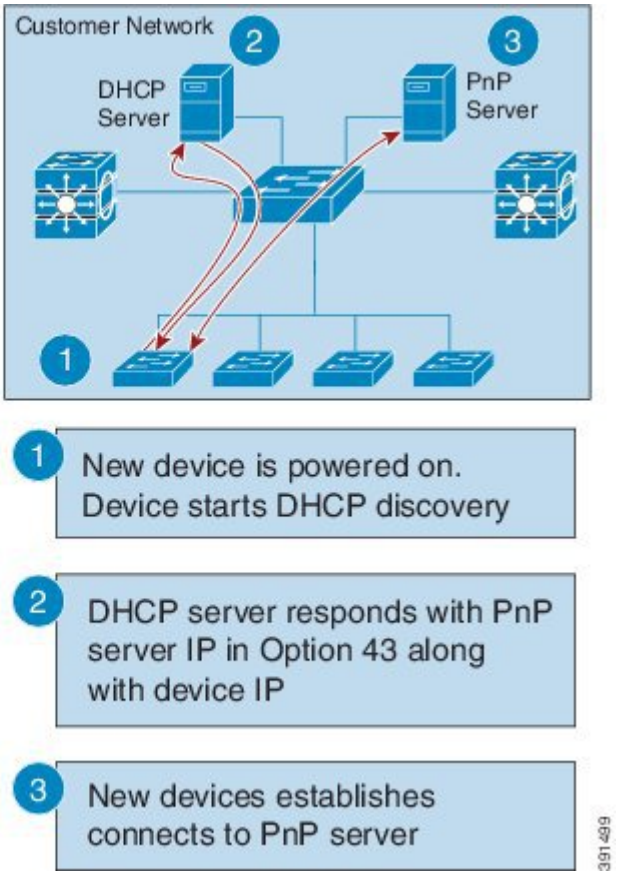
The Cisco Open Plug-n-Play (PnP) agent discovery phase results in acquiring the IP address of the PnP server. When a system boots up without a startup configuration file, the device will not have the IP configuration required to connect to the IP network. In such a scenario, DHCP discovery process is used to obtain the IP parameters needed for the device to have IP connectivity.

When the device boots up, the absence of any startup config on the NVRAM triggers the PnP discovery agent to acquire the IP address of the PnP server. In order to acquire the IP address of the PnP server, the PnP agent goes through one of the following discovery mechanisms:

1. PnP discovery through DHCP server
2. PnP discovery through DHCP snooping
3. PnP discovery through DNS lookup
4. PnP proxy for layer 2 and layer 3 devices
5. PnP deployment application

Plug-n-Play Discovery through DHCP Server
Device with no startup configuration in the NVRAM triggers the Cisco Open Plug-n-Play (PnP) agent to initiate a DHCP discovery process which acquires the IP configuration from the DHCP server required for the device. 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. PnP agent then uses this IP address or hostname to communicate with the PnP server.

Figure 5: DHCP Discovery Process for PnP server



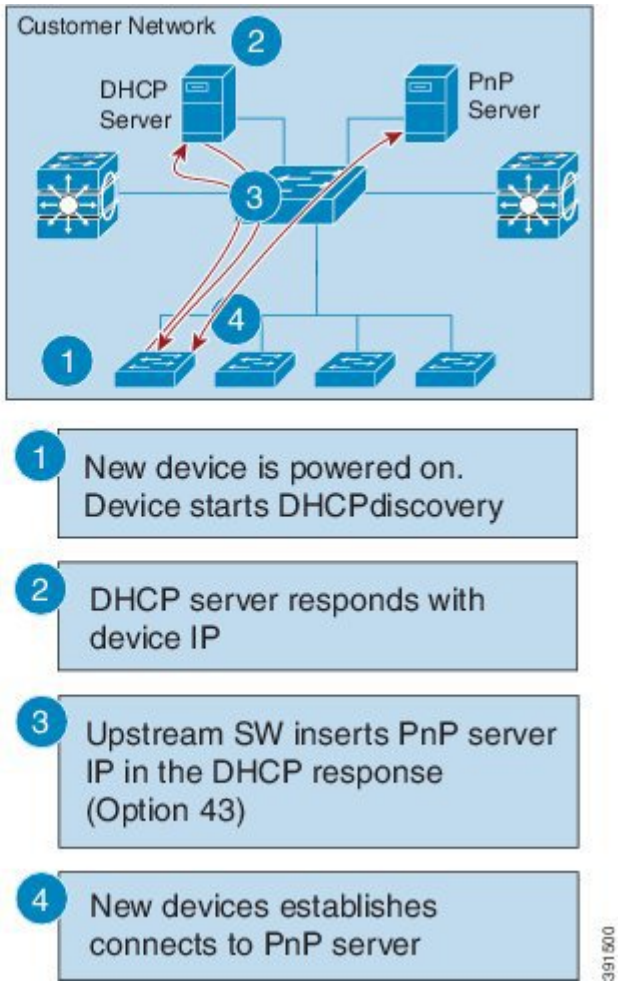
Assumptions:

- New devices can reach DHCP server
- Customer is willing to configure DHCP server for network devices

Plug-n-Play Discovery through DHCP Snooping
If a third party DHCP server cannot be configured to insert any vendor specific options, an existing Cisco Open Plug-n-Play (PnP) enabled device can be configured to snoop in to the DHCP response and insert PnP specific option 43 with the PnP server IP address.

Before inserting the option 43, the snooping agent verifies if the DHCP message is from a Cisco device in the network. The remaining DHCP discovery process is same as described in the previous section.

Figure 6: DHCP Snooping by PnP Server



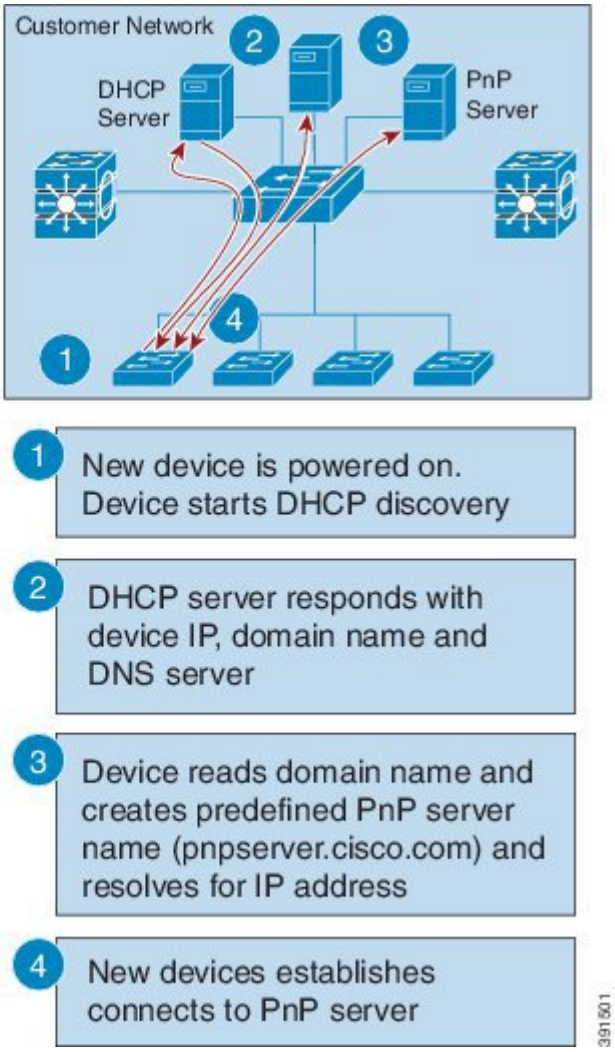
Assumptions:

- New devices can reach DHCP server
- Customer is not willing to configure DHCP server for network devices
- Upstream switch (SW) is configured to snoop DHCP and insert PnP server IP

Plug-n-Play Discovery through DNS Lookup
When the DHCP discovery fails to get the IP address of the Cisco Open Plug-n-Play (PnP) server, the agent falls back on Domain Name System (DNS) lookup method. PnP agent then uses a preset deployment server 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, *deployment.customer.com*. The agent then looks up the local name server and tries to resolve the IP address for the above FQDN.

Figure 7: DNS Lookup for deployment.customer.com



Assumptions:

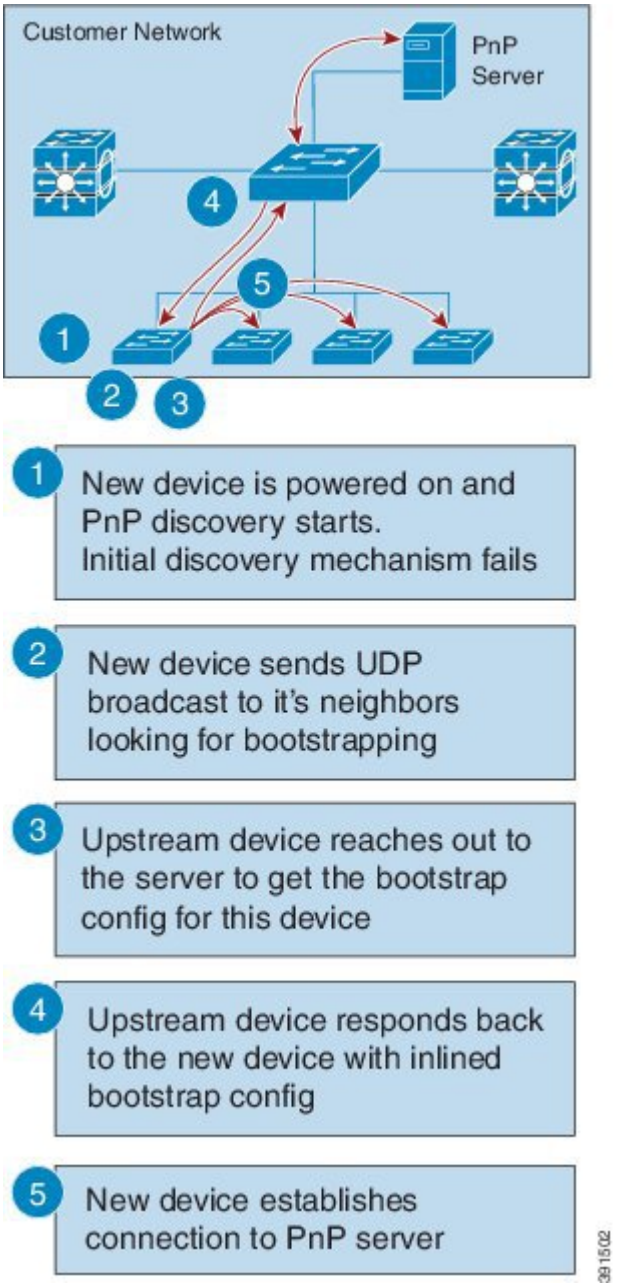
- New devices can reach DHCP server
- Customer deployed PnP server in the network with the name “pnpserver”

Plug-n-Play Proxy Server for Layer 3 and Layer 2 Devices
In the absence of DHCP or Domain Name System (DNS) servers, an existing up and running Cisco Open Plug-n-Play (PnP) enabled device in the neighborhood network can be configured to act as a PnP Neighbor Assisted Provisioning Protocol (NAPP) server.

The NAPP server is part of PnP discovery phase. This server is invoked when the PnP autonomic networking based discovery, DHCP, DNS, Cisco cloud service discovery mechanisms fail to connect to the PnP server.

This device listens to a specific port for any incoming PnP messages. The Cisco device which is trying to come up as a PnP device sends a UDP broadcast message to its network every 30 min for ten times. Hence, if the device does not receive a response, the broadcasts stop after 300 min.

Figure 8: DNS Lookup for Layer 3 and Layer 2 Devices



When the device hosting the proxy server process receives the incoming broadcasts, it verifies the version field in the request and forwards the request to the PnP server if version validation is successful. The proxy server process also caches the unique device identifier (UDI) of the requesting client coming in via incoming datagram before forwarding the request to PnP server.

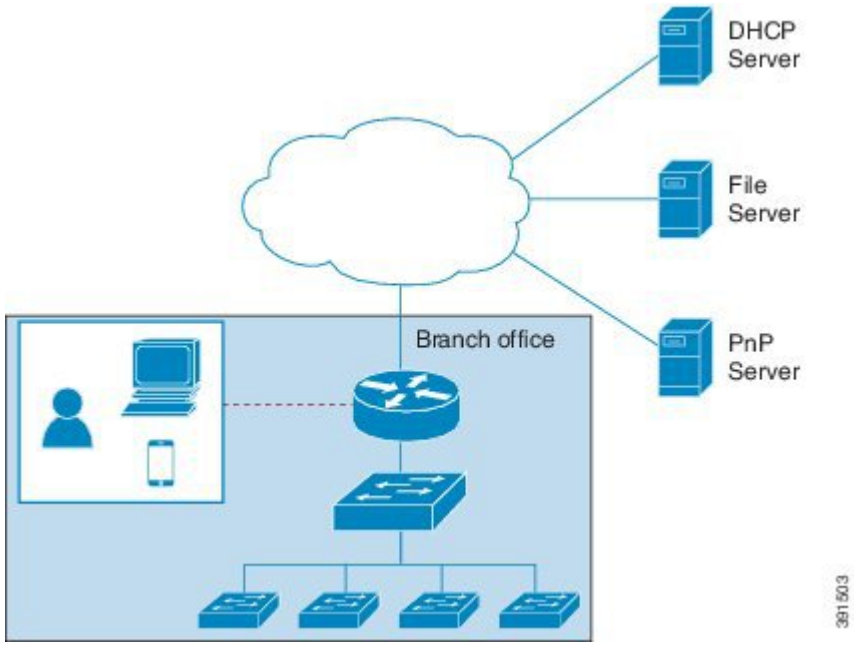
Upon receiving the configlet datagram from PnP server, the proxy server validates UDI in the incoming datagram with the entries in the UDI cache. If validation is successful, proxy server process broadcasts the datagram to a specific port number reserved for the proxy client processes to receive datagrams.

Upon receiving the datagrams, devices running proxy client processes, parse the incoming datagram for the target UDI. If the target UDI in the datagram matches the UDI of the device, proxy client process proceeds with framing, error control and configuring the configlet.

If the target UDI in the datagram fails to match UDI of the device, the packet is dropped.

Plug-n-Play Agent Deployment Application
A Cisco device can alternatively be manually configured by the network administrator using a deployment application running on their PC or on a smart phone. The PC or the smart phone can be connected to the device using an USB or an Ethernet cable.

Figure 9: Manually Configured PnP Agent



Plug-n-Play Agent Deployment Protocol

Deployment can be run over different transports. These transports include Ethernet and IP with Transport Layer Security (TLS). Layer 2 transport is typically used between a deployment agent and a proxy deployment server like a deployment application or as a deployment agent acting as a proxy. Transport between an agent and a server is over an IP connection with TLS for security. Transport between a proxy deployment server and a deployment server is also over IP with TLS.

Plug-n-Play Agent Application Protocol
The Cisco Open Plug-n-Play (PnP) agent application protocol is an XML-based protocol that defines a mechanism that allows network devices to be monitored and controlled by a remote application. The PnP agent is a software module running on a Cisco device. The PnP server is an application running as the network manager that remotely manages the network devices. The main features of the PnP protocol are as follows:

1. Supports HTTP protocols
2. Supports Transport Level Security (TLS) based encryption for HTTP

3 Uses HTTP secure (HTTPS) certificate for TLS handshake

Plug-n-Play Transport over Ethernet
Cisco Open Plug-n-Play (PnP) agent uses the Ethernet based transport in the following two scenarios:

- **Deployment agent communicating with a deployment application on a PC**: In this case, the PC is connected to the device being deployed using an Ethernet cable. The deployment application advertises itself as a deployment server supporting Ethernet transport.
- **Deployment agent communicating with an already deployed device acting as a proxy deployment server**: In this case, the new device being deployed has an Ethernet connection to an already deployed device. The deployment agent on the deployed device responds to the discovery requests and acts as a proxy deployment server for the new device.

Once discovery is complete, the deployment agent starts an unsecured XML stream with the deployment server over Ethernet. This protocol reserves an Ethertype (0xXX TBD) for this purpose. The deployment agent and the server then negotiate to use Extensible Authentication Protocol–Transport Layer Security (EAP-TLS) to protect the communication and complete the EAP-TLS session establishment. The deployment server then authenticates the device with the HTTP secure (HTTPS) certificate or some other supported mechanism.

Plug-n-Play Transport over IP
In Cisco Open Plug-n-Play (PnP) agent, the transport over IP is identical to standard Extensible Messaging and Presence Protocol (XMPP) transport between an XMPP client and an XMPP server. The deployment agent opens TCP connection to the deployment server and starts an XML stream of messages. The server can request the use of Transport Layer Security (TLS) at this time. The agent closes the existing XML stream, initiates a TLS connection to the server, and then restarts the XML stream. The server can request agent authentication over the TLS connection. Refer to RFC 3920 for the transport details for XMPP connections over IP.

Plug-n-Play Agent Security

Security to all Cisco Open Plug-n-Play (PnP) devices is provided at both transport level as well as the application level. The following sections describe the security mechanisms in detail:

Plug-n-Play Transport Layer 3 Security
For non-crypto or non-crypto-enabled images, TLS security choice is not possible. One alternative minimum security is to have the PnP agent initiate the connection to the specified trusted PnP server on port 5222.

Authentication and Authorization between Plug-n-Play Agent and Server
Once the Cisco Open Plug-n-Play (PnP) deployment agent discovers the PnP server, the agent engages the server in a Transport Layer Security (TLS) handshake. In order to authenticate itself to the server, the agent presents its HTTP secure (HTTPS) certificate. The administrator for the PnP server sets device authentication mechanisms which are acceptable for a particular deployment.

The deployment server presents its certificate to the deployment agent so that the agent can authenticate the server. Irrespective of whether the agent is able to verify the server certificate, the agent engages the deployment server in a post-TLS authorization exchange. In this exchange the agent requests the server to present its server authorization token. In response to this request the server presents the authorization token it had obtained from Cisco. The agent verifies the signature on the authorization token. If the authorization token is specific to a Unique Device Identifier (UDI), the agent also ensures its UDI is listed in the authorization-token. At the end of this step, a secure communication channel is established between the deployment agent and the server. This secure communication channel is leveraged by the server to send deployment information to the agent.

Security Methods for the PnP Discovery Process

This section describes the methods that are used to secure the PnP agent-server communication in various scenarios. The security options are used by the PnP agent during the zero-touch PnP server discovery. This section includes the following topics:

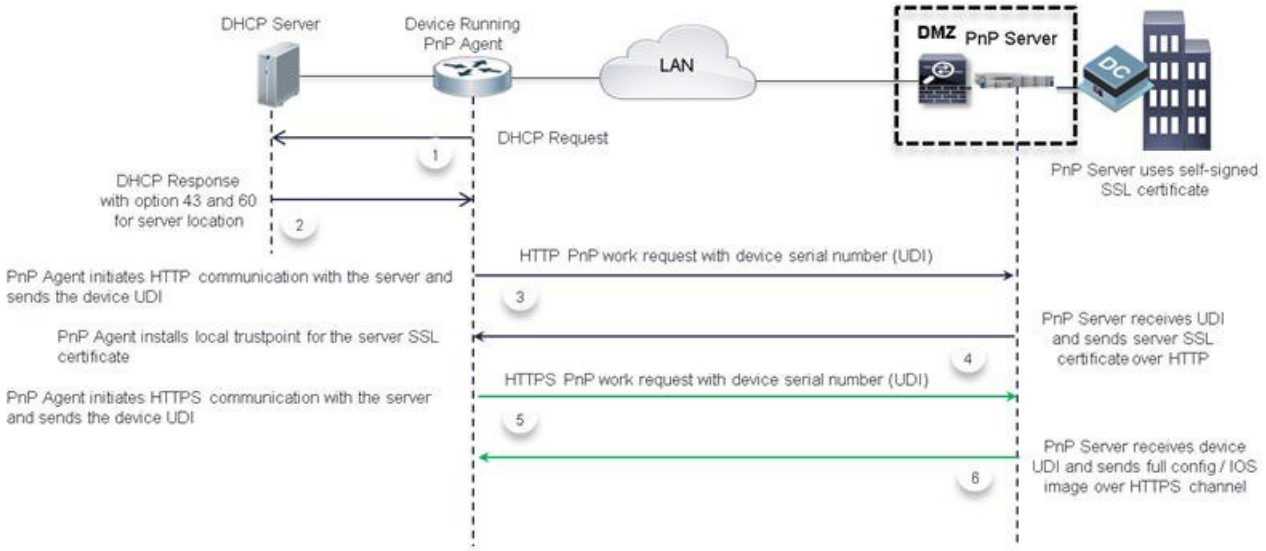
Self-Signed Certificate based Authentication

The PnP server has an option to use a self-signed SSL certificate for server side authentication. When the PnP server uses a self-signed certificate, the PnP discovery cannot be used for automatically initiating secured communication from the agent to the server. The device goes through usual PnP discovery mechanisms and when it finds the server, the agent sends a work-request over HTTP. The server should use the PnP certificate-install service to instruct the agent to install the server self-signed certificate, and then automatically reconnect back to the server over HTTPs.

To keep the solution secured, it is recommended that you use the unsecured port 80 of server to deliver the one-time certificate installation to the devices. All other services should be sent over the secured port.

The following figure shows the end-to-end secured PnP workflow using a self-signed server SSL certificate.

Figure 10: PnP Deployment with Self-Signed Certificate

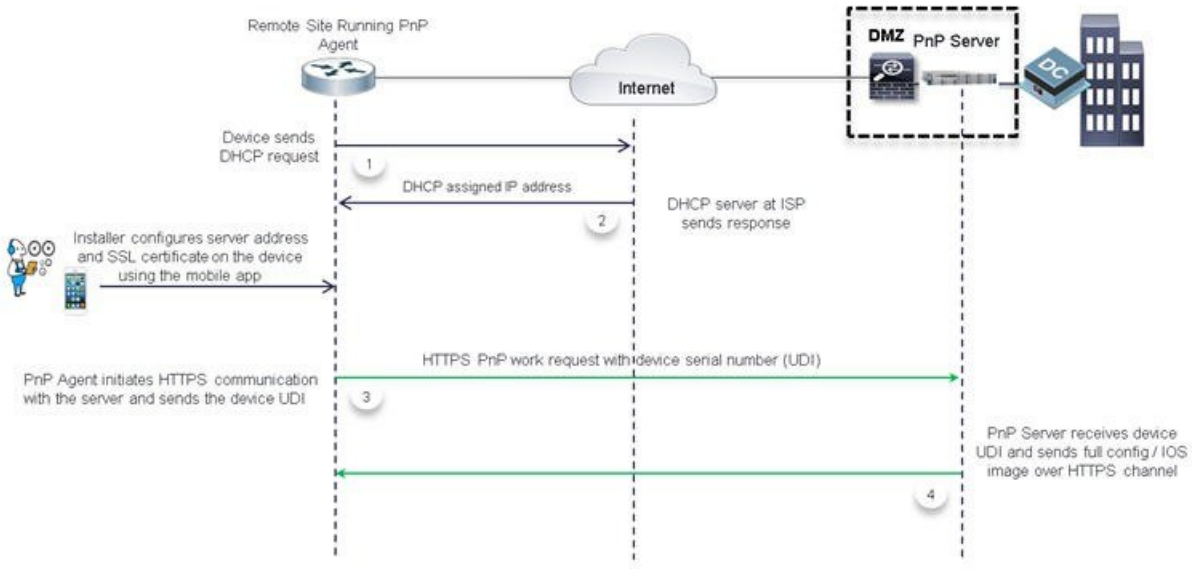


Mobile Device based Secured Installation

As part of this solution, an application for mobile devices is available to configure the bootstrap on the devices. The mobile application can be used to install the server certificate directly on each device along with other bootstrap configuration and then allow the PnP agent initiate secured communication with the server. In this method, the server does not open up any unsecured port for certificate-install.

The following figure shows the end-to-end secured PnP workflow using the application on the mobile devices.

Figure 11: Secured PnP Deployment with the Mobile Application



CA-Signed Certificate based Authentication

Cisco distributes certificates signed by a signing authorities in a tar file format and signs the bundle with Cisco Certificate Authority (CA) signature. This certificate bundle is provided by Cisco infoSec for public downloads on cisco.com.

The certificates from this bundle can be installed on the Cisco IOS device for server side validation during SSL handshake. It is assumed that the server uses a certificate, which is signed by one of the CA that is available in the bundle.

The PnP agent uses the built-in PKI capability to validate the certificate bundle. As the bundle is signed by Cisco CA, the agent is capable of identifying the bundle that is tampered before installing the certificates on the device. After the integrity of the bundle is ensured by the agent, the agent installs the certificates on the device. After the certificates are installed on the device, the PnP agent initiates an HTTPs connection to the server without any additional steps from the server. The following mechanisms helps the PnP agent to initiate a zero-touch secured communication.

DHCP Option-based Discovery

The DHCP option 43 and option 60 is a vendor specific identifier which is used by the PnP agent to locate and connect to the PnP server. To support multiple vendors, the PnP agent in Cisco device sends out a case-sensitive “ciscopnp” as the option 60 string during the DHCP discovery. The DHCP server can be configured with multiple classes matching with a different option 60 string that comes from each network device. After the option 60 string matches, the DHCP server sends out the corresponding option 43 string back to the device. The following is the format for defining the option 43 for PnP deployments:

*option 43 ascii "5A;K5;B2;I10.30.30.10;J443;**Ttftp://10.30.30.10/ ios.p7b**;Z10.30.30.1*

The field 'T' in the PnP string provides an option for the network administrator to specify the location of the certificate bundle, which can be hosted on a local or remote file server.

If the certificate bundle is available at the specified location, then the agent:

1 Downloads the bundle from the file server to the device.

2 Checks the signature of the downloaded bundle to ensure it has a genuine Cisco signature.

3 Installs the certificates on the device.

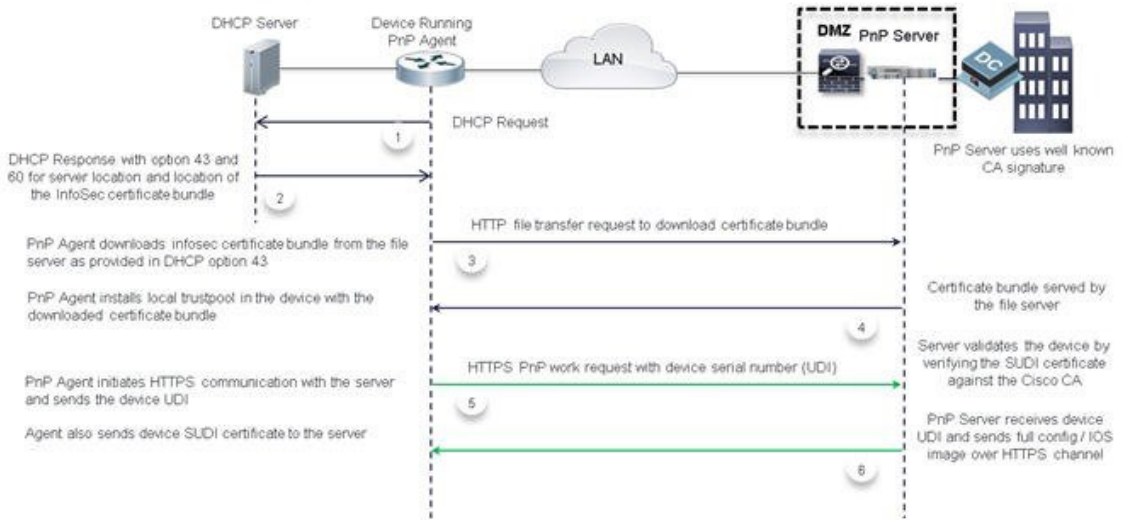
If the 'T' option is not specified and the transport mechanism is specified in the option 43 string as HTTPs, the PnP agent looks for the Cisco signed certificate bundle in the default folder of the same server *http://10.30.30.10:443/certificates/default/cert.p7b* .

If the certificates are available at the default location then the agent performs the steps mentioned above to install the certificates.

After the certificates are installed and the server discovery is complete, the agent initiates the HTTPs connection with the server without any additional configuration. During the HTTPs handshake, the device uses the certificates installed from the bundle to validate the server certificate.

The following figure shows the end-to-end secured PnP workflow using the CA bundle-based certificate.

Figure 12: Secured PnP Deployment with Trustpool



This flow works only if the server is using a certificate signed by one of the known signing authorities that is available in the bundle. If the server uses a certificate that is not a part of the bundle then the HTTPs handshake will fail. When you specify the option 43 string with HTTPs as a transport option and if the bundle download fails, the agent will not fall back to any of the unsecured communication protocol even if the server is reachable. If the transport option is specified as HTTP with a parameter 'T' pointing to a valid certificate bundle location, the agent overrides the transport option HTTP and changes it to HTTPs for secured communication. Generally, the agent will choose the most secured communication from the available options.

The path specified in the DHCP option 43 to locate the certificate bundle file can be an absolute URL or a relative URL. If you specify a relative URL, the agent forms a full URL with the server IP address or hostname as specified in the option 43 string and uses HTTP as the file transfer protocol.

Also, to install the certificates, the agent expects the device to have an updated system clock. Because, you configure the DHCP server first, you cannot specify the current time in the DHCP server. In such a scenario, an IP address or a URL can be specified as

an alternative parameter in the option 43 with the prefix 'Z', which can point the device to a NTP server. The agent synchronizes the clock on the device with the NTP server and then installs the certificates.

DNS-based Discovery

In DNS-based discovery, a DHCP server receives the domain name of the customer network. The domain name is used to create a PnP-specific, fully qualified domain name (FQDN) such as *pnpserver.domain.com*. In this method, the customer network resolves this URL to a valid PnP server IP address. Because, there is no mechanism to specify the certificate location, the agent locates the server certificate to initiate the HTTPs connection without manual intervention.

During the system boot up, the device acquires IP network information from a DHCP server along with the domain name. With the customer specific domain name, the PnP agent creates the following URL pnpserver.domain.com and looks for the Cisco signed certificate bundle in a default folder of the server http://pnpserver.domain.com/certificates/default/cert.p7b.

If the certificate bundle is available at the specified location, then the agent will:

1. Downloads the bundle from the file server to the device.
2. Checks the signature of the downloaded bundle to ensure it has genuine Cisco signature.
3. Installs the certificates on the device.

If the certificate bundle is not available at the specified location, the PnP agent creates a second URL,*pnpcertserver.domain.com* and looks for the Cisco signed certificate bundle in the default folder of the server, *http://pnpcertserver.domain.com/certificates/default/cert.p7b*.

If the certificates are available at the specified location, then the agent performs the steps specified above to install the certificates.

After the certificates are installed and the server discovery is complete, the agent initiates the HTTPs connection with the server at the URL, *pnpserver.domain.com* without any additional configuration. During the HTTPs handshake, the device uses the certificates that are installed from the bundle to validate the server certificate.

Also, to install the certificates, the agent expects the device to have an updated system clock. Because, you configure the DHCP server first, you cannot specify the current time in the DHCP server. In such a scenario, the agent uses a predefined URL, *pnpntpserver.domain.com* which needs to be mapped to a NTP sever to synchronize the clock on the device, and then installs the certificates.

However, if the certificate is not present at either URL, the PnP agent will fall back and establish the HTTP connection to the server using the created FQDN pnpserver.cisco.com. With this workflow, the agent expects the server to use the certificate-install service to install the self-signed certificates first and then start the provisioning steps.

Security Methods for Post-PnP Discovery Process

This section explains the methods provided by the PnP agent which can be, used by the PnP server to secure the client-server communication after the completing the discovery process. This section includes the following topic:

- Certificate Install Service, on page 21

Certificate Install Service

The PnP agent provides a mechanism to manage SSL certificates on the device by providing the certificate-install service to the PnP server. The certificate-install service provides a simple XML to install the server, self-signed or signed by standard CA certificates on the device, before initiating an HTTPs connection. The certificate-install service also provides an option to install the client SSL certificate and instruct the device to use the same SSL certificate during the next device authentication process.

SUDI-based PnP Application Level Authentication

The SSL communication ensures encryption of the data packets exchanged between the server and the device, but does not provide a solution to authenticate the device.

To ensure that the server is talking to a genuine Cisco device, the agent uses the built-in Secure Unique Device Identifier (SUDI) certificate support on the device. SUDI is a X.509 compliant device certificate burnt into the device's secured chip (ACT2) during the manufacture time. The SUDI certificate contains the device's serial number, private-public keys, and the Cisco CA signature. The agent provides the following mechanisms that can be used by the server to authenticate the device as a genuine Cisco device:

- SUDI-based Client Certificate Validation, on page 22
- SUDI-based Serial Number, on page 22

SUDI-based Client Certificate Validation

Before the agent initiates an HTTPs connection with the server, the agent checks whether the device has a built-in SUDI certificate. If the device has a certificate, then the agent sends the SUDI certificate to the client during the SSL handshake for validating. Optionally, the HTTPs server may choose to validate the device using the SUDI certificate during the SSL handshake. After validating, the HTTPs server allows the device to connect to the server. To validate the device's SUDI certificate, the server should use Cisco CA to complete the validation.

SUDI-based Serial Number

If the device is loaded with SUDI certificate, the PnP agent reads the serial number from the SUDI certificate and presents the same information as an additional tag in the work-request body for all communication with the server. To achieve this, the following optional tag is added in the work-info message, which goes out from the device in every work-request. This field is optional and does not show up for devices that does not have SUDI certificate.

```
<xs:element name="deviceId" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="udi" src="xs:string" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="hostname" minOccurs="0" maxOccurs="1" type="xs:string"/>
<xs:element name="SUDI" minOccurs="0" maxOccurs="1" type="xs:string"/>
<xs:element name="authRequired" minOccurs="1" maxOccurs="1" type="xs:boolean"/>
<xs:element name="viaProxy" minOccurs="0" maxOccurs="1" type="xs:boolean"/>
<xs:element name="securityAdvise" minOccurs="0" maxOccurs="1" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

There is no change in the existing UDI mechanism that is read from the chassis inventory. The agent continues to be backwards compatible by sending the chassis UDI as the primary identifier. The server can use the additionally provided SUDI-based serial number to authenticate the device and then continue to use the primary UDI. For the devices without a SUDI certificate, the agent does not send this additional SUDI-based serial number. Therefore, the server should continue with the primary UDI for authentication and further communication.

There is no mechanism available to read the SUDI-based serial number from member hardware and there is no change in how UDI is read from other members on a stack or HA unit. The agent will continue to read the UDI from all the hardware units as it does presently.

SUDI-Based Device Authentication

In SUDI-based device authentication, the agent checks whether the device has a built-in SUDI certificate at the boot-up time. If the device is loaded with the SUDI certificate, the agent provides a new PnP service, which allows the server to help the device to identify

itself. The availability of this new service depends on the presence of the SUDI certificate and is listed in the agent's capability service. The new capability response has the following response:

```
<xs:element name="supported-capabilities" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="Capability" maxOccurs="unbounded">
<xs:simpleType>
<xs:restriction base="xs:string">
     <xs:enumeration value="cisco:pnp:capability"/>
     <xs:enumeration value="cisco:pnp:device-auth"/>
     <xs:enumeration value="cisco:pnp:cli-config"/>
     <xs:enumeration value="cisco:pnp:cli-exec"/>
     <xs:enumeration value="cisco:pnp:config-upgrade"/>
     <xs:enumeration value="cisco:pnp:device-info"/>
     <xs:enumeration value="cisco:pnp:file-transfer"/>
     <xs:enumeration value="cisco:pnp:image-install"/>
     <xs:enumeration value="cisco:pnp:reload"/>
     <xs:enumeration value="cisco:pnp:snmp"/>
     <xs:enumeration value="cisco:pnp:syslog"/>
     <xs:enumeration value="cisco:pnp:snooping"/>
     <xs:enumeration value="cisco:pnp:napp"/>
     <xs:enumeration value="cisco:pnp:location"/>
     <xs:enumeration value="cisco:pnp:script-exec"/>
     <xs:enumeration value="cisco:pnp:licensing"/>
     <xs:enumeration value="cisco:pnp:device-id"/>
     <xs:enumeration value="cisco:pnp:backoff"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Along with the above change in the capability-service, the agent adds an additional field under the hardware-info section of the device-info response, to specify and check whether the SUDI certificate is built into the device.

```
<xs:element name="hardwareInfo" minOccurs="0" maxOccurs="1"><!-- Only if
harwadre-info type is requested -->
<xs:complexType>
<xs:sequence>
<xs:element name="hostname" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="vendor" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="sudiSupported" minOccurs="0" maxOccurs="1" type="xs:boolean"/>
<xs:element name="platformName" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="processorType" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="hwRevision" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="mainMemSize" minOccurs="1" maxOccurs="1" type="xs:integer"/>
<xs:element name="ioMemSize" minOccurs="1" maxOccurs="1" type="xs:integer"/>
<xs:element name="boardId" minOccurs="1" maxOccurs="1" type="xs:integer"/>
<xs:element name="boardReworkId" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="processorRev" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="midplaneVersion" minOccurs="1" maxOccurs="1" type="xs:string"/>
<xs:element name="location" minOccurs="1" maxOccurs="1" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

After, the agent initiates an HTTPs connection with the server and sends a work-request, the server should be able to use the device authentication service for a challenge request-response. The device authentication service requires a minimum of one field to generated a string by the server. Optionally, the server can send a list of encryptions and hashing methods that it can support. The agent checks whether it has the capability to use any of the listed encryption methods specified by the server, uses the encryption method and sends a notification to the server. If the agent does not have the capability to use any of the methods specified by the server, then the agent responds with an error message.

```
<xs:schema xmlns="urn:cisco:pnp:device-auth">
<xs:element name="request">
<xs:element name="hash-methods" type="xs:string" minOccurs="1" maxOccurs="n"/>
<xs:element name="encryption-methods" type="xs:string" minOccurs="1" maxOccurs="n"/>
<xs:element name="challenge-string" type="xs:string" minOccurs="1" maxOccurs="1"/>
</xs:element>
</xs:schema>
```

When the server sends a device authentication service request to the agent, the agent does the following:

1. Uses one of the specified encryption and hashing methods.
2. If the agent does not have capability to use one of the specified encryption and hashing methods, the agent responds with an error message.
3. Encrypts the challenge string provided by the server using the private key using the PKI APIs.
4. Sends a response back with the following:
    1. Cipher text
    2. Methods used to cipher
    3. Certificate (SUDI or client installed certificate)

The following is the response schema:

```
<xs:schema xmlns="urn:cisco:pnp:device-auth">
<xs:element name="response">
<xs:element name="hash-used" type=”xs:string” minOccurs="1" maxOccurs="1"/>
<xs:element name="cipher-used" type=”xs:string” minOccurs="1" maxOccurs="1"/>
<xs:element name="cipher-string" type=”xs:string” minOccurs="1" maxOccurs="1"/>
<xs:element name="certificate" type=”xs:string” minOccurs="1" maxOccurs="1"/>
</xs:element>
</xs:schema>
```

After, the server receives the above response from the device, the server does the following:

1. Verifies the SUDI or the client certificate against the Cisco or customer CA.
2. Decrypts the cipher-string using the public key that is available in the SUDI or client certificate.
3. Verifies whether the deciphered string matches the original version.
4. Generates a session key (string) and sends it back to the device as an acknowledgment.

```
<xs:element name="workInfo" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
    <xs:element name="session-key" type="xs:string" fixed="" minOccurs="0" maxOccurs="1"/>
    <xs:element name="bye" type="xs:string" fixed="" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

After the agent receives the final acknowledgment from the server with the session-key, it associates the corresponding profile with the provided session-key and sends it to the server as an attribute in the root PnP section of all the subsequent messages that the agent sends.

The following example shows the session-key in the work information service:

```
<xs:schema xmlns="urn:cisco:pnp" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:svc="urn:cisco:pnp:work-info" targetNamespace="urn:cisco:pnp">
<xs:import namespace="urn:cisco:pnp:device-id" schemaLocation="pnp_device_id_body.xsd"/>
<xs:import namespace="urn:cisco:pnp:work-info"
schemaLocation="pnp_device_work_info_body.xsd"/>
<xs:element name="pnp">
<xs:complexType>
<xs:sequence>
<xs:element ref="svc:info" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
<xs:attribute name="version" fixed="1.0" use="required"/>
<xs:attribute name="udi" type="xs:string" use="required"/>
<xs:attribute name="session-key" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
```

```
</xs:schema>
```

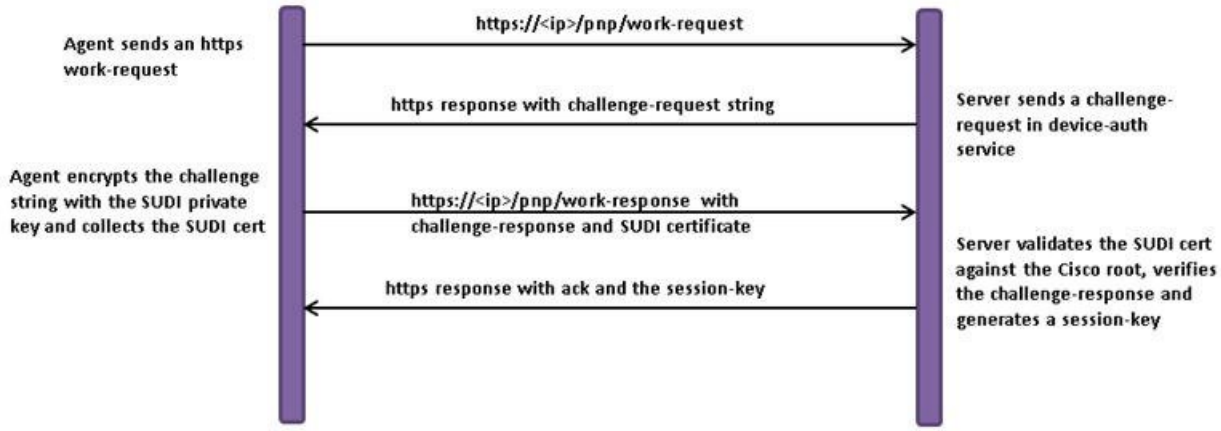
The following example shows the session-key in the response section of image-install service. The session-key appears in the response section of all other services including the notify-services.

```
<xs:schema xmlns="urn:cisco:pnp" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:svc="urn:cisco:pnp:image-install" targetNamespace="urn:cisco:pnp">
 <xs:import namespace="urn:cisco:pnp:image-install"
schemaLocation="pnp_image_install_body.xsd"/>
 <xs:simpleType name="NoCheckTime">
  <xs:restriction base="xs:positiveInteger">
   <xs:minInclusive value="1"/>
   <xs:maxInclusive value="180"/>
  </xs:restriction>
 </xs:simpleType>
        <xs:element name="pnp">
                <xs:complexType>
                        <xs:choice>
                                <xs:element ref="svc:request" minOccurs="1" maxOccurs="1"/>
                                <xs:element ref="svc:response" minOccurs="1" maxOccurs="1"/>
                      </xs:choice>
                      <xs:attribute name="version" type="xs:string" fixed="1.0" use="required"/>
                      <xs:attribute name="udi" type="xs:string" use="required"/>
                      <xs:attribute name="session-key" type="xs:string" use="optional"/>
                      <!-- noCheckTime in minutes, minimum 1, maximum 180 -->
                      <xs:attribute name="noCheckTime" type="NoCheckTime" use="optional"/>
                      <xs:attribute name="postReloadPriv" type="xs:string" fixed="ztd"
                      use="optional"/>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

The server validates the session-key before sending any message from the device. Optionally, the server maintains a timer for the session-keys and moves to invalid status when the timer expires. If the agent sends a message with an expired session-key, the server repeats the device authentication process and generate a new session-key before sending to the same device again. If the device sends a request without any session-key, then the server performs the device authentication process and generates a new session-key before sending to the same device.

The following figure displays the message sequence between the agent and the server to accomplish the device authentication using the SUDI certificate.

Figure 13: Message Sequence



How to Configure Open Plug-n-Play Agent

Configuring Open Plug-n-Play Agent Profile

Perform the following task to create an Open Plug-n-Play (PnP) agent profile:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Open Plug-n-Play Agent Device

Perform the following task to create an Open Plug-n-Play (PnP) agent device:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**device** {**username** *username* } {**password {0 | 7}** *password*} **Example:** `Device(config-pnp-init)# device username sjohn password 0 Tan123`	Configures the PnP agent on the device. • Establishes a username and password based authentication system. • *username*—User ID • *password*—Password that a user enters • **0**—Specifies that an unencrypted password or secret (depending on the configuration) follows. • **7**—Specifies that an encrypted (hidden) password follows.
Step 5	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Open Plug-n-Play Reconnect Factor

Perform the following task to configure the time to wait before attempting to reconnect a session in either fixed-interval-backoff, exponential-backoff, or random-exponential-backoff mode:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**reconnect** [*pause-time* [*exponential-backoff-factor* [**random**]]] **Example:** `Device(config-pnp-init)# reconnect 100 2 random`	Specifies the time for the PnP agent initiator profile to wait before attempting to reconnect a session. • The pause-time value is the time to wait, in seconds, before attempting to reconnect after a connection is lost. The range is from 1 to 2000000. The default is 60. • Exponential backoff factor value is the value that triggers the reconnect attempt exponentially. The range is from 2 to 9.
Step 5	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Open Plug-n-Play HTTP Transport Profile

Perform the following task to create a HTTP transport profile of the Open Plug-n-Play (PnP) agent manually on a device.

Both IPv4 and IPv6 addresses can be used for PnP server IP configuration. Alternately, a hostname can also be used in the configuration to connect to the PnP server.

Every profile can have one primary server and a backup server configuration. The PnP agent attempts to initiate a connection with the primary server first and if it fails, it will try the backup server. If the backup server fails, the PnP agent will attempt to connect to the primary server again. This will continue until a connection is established with one of the servers.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**transport http host** *host-name* [**port** *port-number*] [**source** *interface-type*] **Example:** `Device(config-pnp-init)# transport http host hostname-1 port 1 source gigabitEthernet 0/0/0`	Creates a HTTP transport configuration for the PnP agent profile based on the hostname of the server on which the PnP agent is deployed. • The value of the host specifies the host name, port, and source of the server. • The value of the port-number specifies the port that is used. • The value of the interface-type specifies the interface on which the agent is connected to the server.
Step 5	**transport http ipv4** *ipv4-address* [**port** *port-number*] [**source** *interface-type*] **Example:** `Device(config-pnp-init)# transport http ipv4 10.0.1.0 port 221 source gigabitEthernet 0/0/0`	Creates a HTTP transport configuration for the PnP agent profile based on the IPv4 address of the server on which the PnP agent is deployed.
Step 6	**transport http ipv6** *ipv6-address* [**port** *port-number*] [**source** *interface-type interface-number*] **Example:** `Device(config-pnp-init)# transport http ipv6 2001:DB8:1::1 port 331 source gigabitEthernet 0/0/1`	Creates a HTTP transport configuration for the PnP agent profile based on the IPv6 address of the server on which the PnP agent is deployed.
Step 7	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Open Plug-n-Play HTTPS Transport Profile

Perform the following task to create a HTTP Secure (HTTPS) transport profile of the Open Plug-n-Play (PnP) agent manually on a device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**transport https host** *host-name* [**port** *port-number*][**source** *interface-type*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# transport https host example.com port 231 source gigabitEthernet 0/0/0 localcert abc remotecert xyz`	Creates a HTTPS transport configuration for the PnP agent profile based on the hostname of the server on which the PnP agent is deployed. • The value of *localcert* specifies the trustpoint used for client-side authentication during the transport layer security (TLS) handshake. • The value of *remotecert* specifies the trustpoint used for server certificate validation. **Note** Configure the trustpoint-name using the **crypto pki trustpoint** command.
Step 5	**transport https ipv4** *ipv4-address* [**port** *port-number*][**source** *interface-type*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# transport https ipv4 10.0.1.0 port 221 source gigabitEthernet 0/0/0 localcert abc remotecert xyz`	Creates a HTTPS transport configuration for the PnP agent profile based on the IPv4 address of the server on which the PnP agent is deployed.

	Command or Action	Purpose
Step 6	**transport https ipv6** *ipv6-address* [**port** *port-number*][**source** *interface-type interface-number*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# transport https ipv6 2001:DB8:1::1 port 331 source gigabitEthernet 0/0/1 localcert abc remotecert xyz`	Creates a HTTPS transport configuration for the PnP agent profile based on the IPv6 address of the server on which the PnP agent is deployed.
Step 7	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Backup Open Plug-n-Play Device

Perform the following task to create a backup profile and to enable or disable Open Plug-n-Play agent manually on a device:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**backup device** {**username** *username* } {**password {0 | 7}** *password*} **Example:** `Device(config-pnp-init)# backup device username sjohn password 0 Tan123`	Configures the PnP agent backup profile on the device. • Establishes a username and password based authentication system. • *username*-User ID • *password*-Password that a user enters

	Command or Action	Purpose
		• 0—Specifies that an unencrypted password or secret (depending on the configuration) follows. • 7—Specifies that a hidden password follows.
Step 5	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Backup Open Plug-n-Play Reconnect Factor

Perform the following task to configure backup reconnection of the Open Plug-n-Play (PnP) agent to the server in either fixed-interval-backoff, exponential-backoff, or random-exponential-backoff manner :

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**backup reconnect** [*pause-time* [*exponential-backoff-factor* [**random**]]] **Example:** `Device(config-pnp-init)# backup reconnect 100 2 random`	Specifies the time for the PnP agent initiator profile to wait before attempting to reconnect a session. • The pause-time value is the time to wait, in seconds, before attempting to reconnect after a connection is lost. The range is from 1 to 2000000. The default is 60. • Exponential backoff factor value is the value that triggers the reconnect attempt exponentially. The range is from 2 to 9.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Backup Open Plug-n-Play HTTP Transport Profile

Perform the following task to create a backup HTTP transport profile of the Open Plug-n-Play (PnP) agent manually on a device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.
Step 4	**backup transport http host** *host-name* [**port** *port-number*] [**source** *interface-type*] **Example:** `Device(config-pnp-init)# backup transport http host hostname-1 port 1 source gigabitEthernet 0/0/0`	Creates a backup HTTP transport configuration for the PnP agent profile based on the hostname of the server on which the PnP agent is deployed. • The value of the host specifies the host name, port, and source of the server. • The value of the port-number specifies the port that is used. • The value of the interface-type specifies the interface on which the agent is connected to the server.

	Command or Action	Purpose
Step 5	**backup transport http ipv4** *ipv4-address* [**port** *port-number*] [**source** *interface-type*] **Example:** `Device(config-pnp-init)# backup transport http ipv4 10.0.1.0 port 221 source gigabitEthernet 0/0/0`	Creates a backup HTTP transport configuration for the PnP agent profile based on the IPv4 address of the server on which the PnP agent is deployed.
Step 6	**backup transport http ipv6** *ipv6-address* [**port** *port-number*] [**source** *interface-type interface-number*] **Example:** `Device(config-pnp-init)# backup transport http ipv6 2001:DB8:1::1 port 331 source gigabitEthernet 0/0/1`	Creates a backup HTTP transport configuration for the PnP agent profile based on the IPv6 address of the server on which the PnP agent is deployed.
Step 7	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Backup Open Plug-n-Play HTTPS Transport Profile

Perform the following task to create a backup HTTPS transport profile of the Open Plug-n-Play (PnP) agent manually on a device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp profile** *profile-name* **Example:** `Device(config)# pnp profile test-profile-1`	Creates a PnP agent profile and enters the PnP profile initialization mode. • String of alphanumeric characters that specify a name for the PnP agent profile. Profile names cannot be duplicated.

	Command or Action	Purpose
Step 4	**backup transport https host** *host-name* [**port** *port-number*][**source** *interface-type*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# backup transport https host example.com port 231 source gigabitEthernet 0/0/0 localcert abc remotecert xyz`	Creates a HTTPS backup transport configuration for the PnP agent profile based on the hostname of the server on which the PnP agent is deployed. • The value of *localcert* specifies the trustpoint used for client-side authentication during the transport layer security (TLS) handshake. • The value of *remotecert* specifies the trustpoint used for server certificate validation.
Step 5	**backup transport https ipv4** *ipv4-address* [**port** *port-number*][**source** *interface-type*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# backup transport https ipv4 10.0.1.0 port 221 source gigabitEthernet 0/0/0 localcert abc remotecert xyz`	Creates a HTTPS backup transport configuration for the PnP agent profile based on the IPv4 address of the server on which the PnP agent is deployed.
Step 6	**backup transport https ipv6** *ipv6-address* [**port** *port-number*][**source** *interface-type interface-number*][**localcert** *trustpoint-name*][**remotecert** *trustpoint-name*] **Example:** `Device(config-pnp-init)# backup transport https ipv6 2001:DB8:1::1 port 331 source gigabitEthernet 0/0/1 localcert abc remotecert xyz`	Creates a HTTPS backup transport configuration for the PnP agent profile based on the IPv6 address of the server on which the PnP agent is deployed.
Step 7	**end** **Example:** `Device(config-pnp-init)# end`	Exits the PnP profile initialization mode and returns to privileged EXEC mode.

Configuring Open Plug-n-Play Agent Tag

Perform the following task to create an Open Plug-n-Play (PnP) agent tag information:

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**pnp tag** *tag-name* **Example:** `Device(config)# pnp tag xyz`	Use the **pnp tag** command to configure the tag for the device. The neighboring Cisco devices will receive this tag information through Cisco Discovery Protocol (CDP). **Note** If there is an existing tag for the device, the tag name can be only changed when the xml schema is sent by the PnP server to change the tag name. The tag name cannot be overwritten. • String of alphanumeric characters that specify a name for the PnP agent tag.
Step 4	**end** **Example:** `Device(config)# end`	Exits the global configuration mode and returns to privileged EXEC mode.

Trouble Shooting and Debugging

To run the debugging on the Open Plug-n-Play (PnP) server, start the server, configure the PnP profile and PnP transport. That is, start the service interaction between PnP agent and PnP server.

Capture the debugs by executing the **debug pnp service** command.

Glossary

NAPP Server: The Cisco Open Plug-n-Play (PnP) Neighbor Assisted Provisioning Protocol (NAPP) server that is part of PnP discovery phase. This is a PnP enabled device in the neighborhood that can be configured to act as a PnP proxy server.

PnP Agent: An embedded agent on the device to automate deployment process

PnP Helper Applications: Applications on smart phones and personal computers that facilitate deployment. PnP helper applications are not specific to a customer or device and can be used in any deployment scenario. May be needed in limited scenarios

PnP Protocol: Protocol between the PnP agent and PnP server. This is an open protocol allowing third-party development of PnP servers

PnP Server: A central server that manages and distributes deployment information (images, configurations, files, and licenses) for the devices being deployed. Open Plug-n-Play (PnP) server provides a north bound interface for management applications and communicates with the PnP agents on the devices using the PnP protocol.

Additional References for Open Plug-n-Play Agent

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
PnP commands: Complete command syntax, command mode, command history, defaults, usage guidelines, and examples	Cisco IOS PnP Command Reference
Cisco Network Plug and Play solution	Solution Guide for Cisco Network Plug and Play.
How to use the Cisco Network Plug and Play in the APIC-EM to configure Cisco network devices.	Configuration Guide for Cisco Network Plug and Play on Cisco APIC-EM.
How to deploy the APIC-EM.	Cisco Application Policy Infrastructure Controller Enterprise Module Deployment Guide.
Getting started with the APIC-EM.	Cisco APIC-EM Quick Start Guide.

MIBs

MIB	MIBs Link
• CISCO-BULK-FILE-MIB • CISCO-DATA-COLLECTION-MIB • CISCO-PROCESS-MIB • Expression-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Open Plug-n-Play Agent

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for Open Plug-n-Play Agent

Feature Name	Releases	Feature Information
Open Plug-n-Play Agent		The Cisco Open Plug-n-Play agent converges existing solutions into a unified agent and adds functionality to enhance the current deployment solutions. The following commands were introduced or modified by this feature: **backup device**, **backup profile**, **backup reconnect**, **backup transport http**, **backup transport https**, **backup transport xmpp socket**, **backup transport xmpp starttls**, **backup transport xmpp tls**, **device**, **pnp**, **profile**, **reconnect (pnp)**, **transport http**, **transport https**, **transport xmpp socket**, **transport xmpp starttls**, and **transport xmpp tls**.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA 95134-1706
USA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands