LTRCRS-2017

Catalyst 9000 Switching Innovation Lab

Dimitar Hristov - TME
Jay Sharma - TME

Introduction Or Learning Objectives

Upon completion of this lab you, you will be able to:

- Configure Bonjour via DNAC ~30 mins
- Configure Flexlinks + as STP alternative ~15 mins
- Configure MUD IoT classification via ISE ~15 mins
- Configure ZTP provisioning ~15 mins
- Configure 90W based on 802.3bt
    - Use IGOR LED lights powered by 802.3bt PoE ports ~15 mins
- Configure App hosting
    - Configure latest innovations into App Hosting Framework ~15 mins
- Demonstrate xFSU on Catalyst 9300 only
    - Show how xFSU improve the upgrade time ~15 mins

The Cisco Catalyst 9000 Switches are the next generation of enterprise-class switches built for security, Internet of Things (IoT), mobility, and multicloud running on the feature-rich Cisco IOS-XE and programmable Unified Access DataPlane (UADP) ASIC technologies.

This lab intends to bring to you a hands-on experience of some of the newer capabilities of the Catalyst 9000 family of switches. In the lab you will use both CLI and DNA Center orchestration to experience these new innovations.

Most of the lab scenarios are independent of each another, with any exceptions noted in the guide. As there may not be enough time to finish all of the scenarios, please pick the most important topics to complete first.

Disclaimer

This training document is to familiarize with some of Catalyst 9000 switches innovations. Although the lab design and configuration examples could be used as a reference, it's not a real design, thus not all recommended features are used, or enabled optimally. For the design related questions please contact your representative at Cisco, or a Cisco partner.

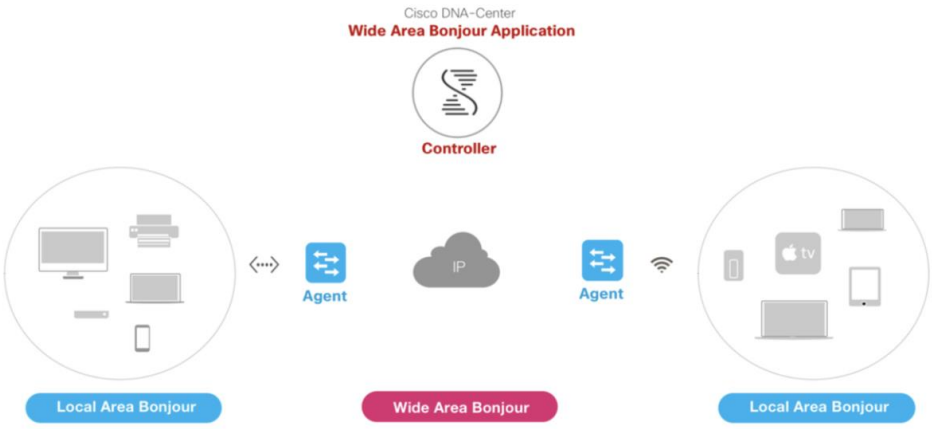
Scenario Cisco DNA Service for Bonjour

In this lab activity, you will learn what Bonjour is and how Cisco DNA Center along with Cisco Catalyst 9000 Series Switches can be used to scale Bonjour on challenging and complex enterprise networks. This feature was introduced with Catalyst 9000 Series Switches with IOS-XE 16.11.1 and Cisco DNA Center 1.3.1 releases.

The Apple Bonjour protocol is a zero-configuration solution enabling plug-n-play communication between connected devices to share and access content, and to discover and manage devices. The Bonjour technology is an industry standard, zero-configuration protocol designed for single Layer 2 domains, which is ideal for small, flat, single-domain setups, such as home and small-business network environments.

Enterprise administrators face several challenges in large and complex Enterprise networks to seamlessly introduce a Bonjour technology that is originally designed to operate in a single Layer 2 broadcast domain. Due to the proliferation of Bonjour devices and mandatory service requirements, networking vendors introduced a gateway solution that allows service discovery between local network segments. The solution overcomes the initial challenge but continues to be limited as the service discovery and distribution support only a single gateway, without any end-to-end solution. The centralized architecture of a single gateway quickly becomes a bottle neck as the network expands, demanding more scale and performance.

Cisco Digital Network Architecture (DNA) Service for Bonjour is a solution that comprises two core components - Controller and Agent - that are designed to precisely address Enterprise network challenges. Cisco DNA Center introduces a new Cisco Wide Area Bonjour Application that network administrators can download from the application catalogue server to activate the Bonjour Controller function in DNA Center. Cisco IOS-XE software on network devices introduces a new and advanced Wide Area Service Discovery Gateway (SDG) function that performs the Agent role in the overall solution



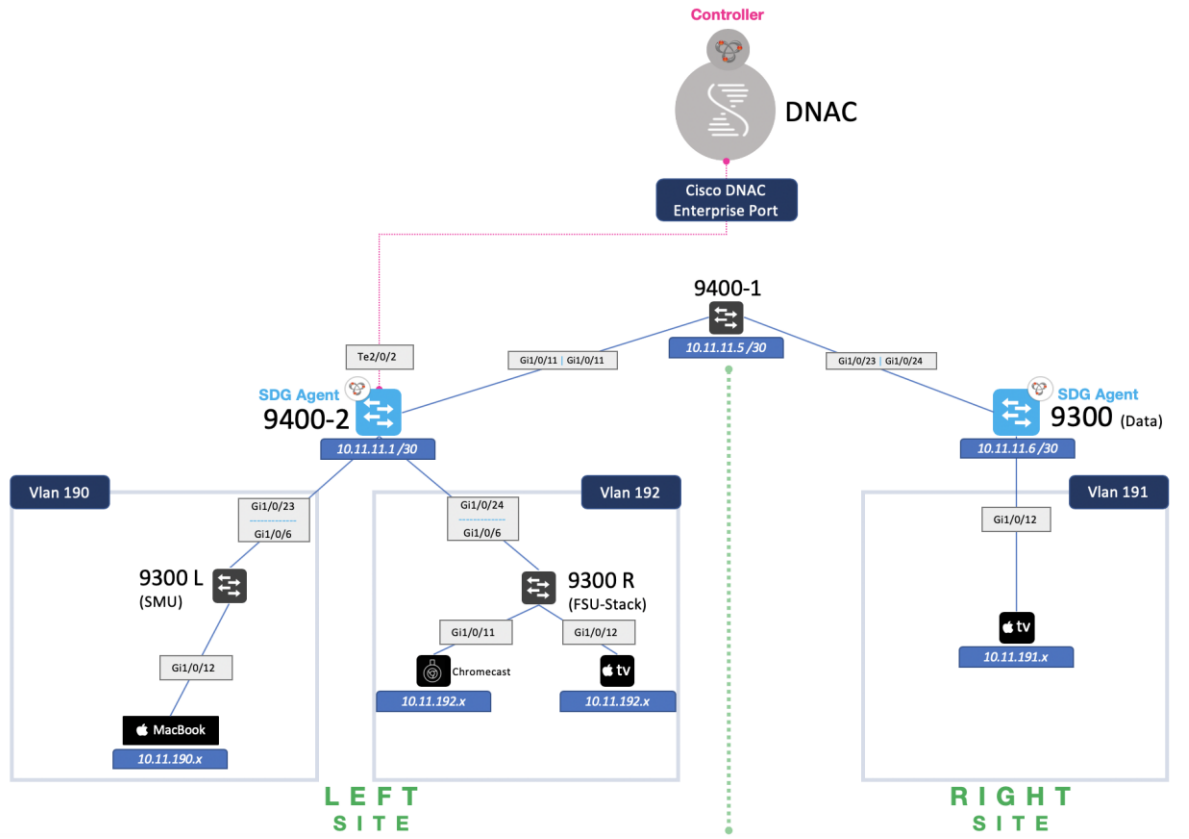
The figure above illustrates a reference architecture of Cisco DNA Service for Bonjour supporting a two-tier, end-to-end distributed architecture.

Solution Components

The DNA Service for Bonjour solution is an end-to-end solution that includes the following key components:

- Cisco DNA Center Appliance – The Cisco DNA Center Appliance is the foundational controller and analytics platform of the Cisco Digital Network Architecture. The Cisco DNA Center Appliance can be deployed in single standalone, non-redundant mode or in Cisco recommended cluster mode providing best-in-class system, application and service redundancy.
- Cisco Wide Area Bonjour Application – The Cisco Wide Area Bonjour (WAB) Application is an add-on service that operates on the Cisco DNA Center Appliance that can be deployed in datacenter. The WAB Application enables the Bonjour Controller function and builds network-wide secure communication channels with trusted SDG Agents for global centralized services management and controlled service routing.
- Cisco SDG Agent – The Cisco Catalyst 9000 Series Switch or an ISR 4000 series router functions as an SDG Agent and communicates with the Bonjour Service endpoints within the Layer 2 domain and central Cisco DNA Center controller.
- Endpoints – A Bonjour endpoint is any device following RFC 6762 standards that advertises or browses Bonjour services. The Bonjour endpoints can be eithers LANs or WLANs. The Wide Area Bonjour application is designed to integrate with known Bonjour services, including AirPlay, Google Chromecast, and AirPrint.

Network Diagram DNAC Bonjour



Task 1: Configure Local Area Bonjour domain

Every Lab Pod includes a complete setup that simulates a real world enterprise setting and contains all the solution components as shown in the above diagram.

Step 1: Configure underlay switches

Local-Area SDG Domain - The Cisco Catalyst 9000 switches at the Layer 3 boundary function as Service Discovery Gateways (SDGs) for local cache discovery and distribution functions between local VLANs. In this controller-less Bonjour solution, the SDG gateway switch provides a single gateway solution at the LAN Distribution block. The SDG switch communicates with local Bonjour endpoints to build and manage the services information. The Bonjour gateway function is ineffective between Bonjour endpoints in same Layer 2 network as they follow standards-based flood-n-learn rules.

As part of the service implementation, it is imperative to understand policy enforcement points to ensure service routing is performed based on policies. The following sub-

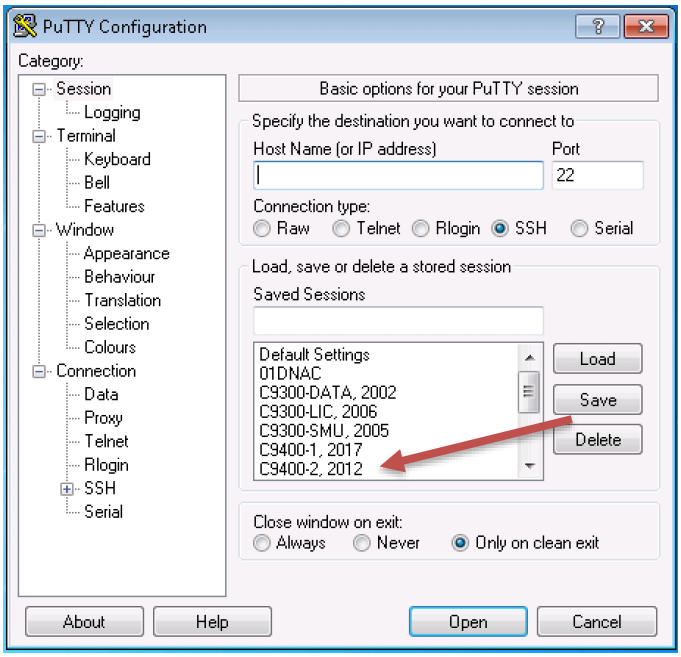
section describes the function of each policy type in Local Area domains:

- Local Area Bonjour – Ingress Policy – This mandatory policy is applied on the Cisco SDG device on the VLAN interface to permit Bonjour services from providers, such as Printers, or requests from receivers. The SDG device only processes whitelisted Bonjour services while an implicit deny rule at the end of list drops anything not explicitly permitted.

- Local Area Bonjour – Egress Policy – This optional egress policy is applied on Cisco SDG devices on a VLAN interface to permit outbound Bonjour responses to discovery requests from receiver endpoints. The SDG device may provide Bonjour service responses from the local cache to requesting devices on other VLAN interfaces if matching services are found. This Local Area egress service policy is only applicable for local cache responses, it does not enforce trusted and validated responses for Wide Area Bonjour remote services received from Cisco DNA-Center.

In this lab, we have already configured the underlay of the network. For our Local Area Bonjour setup, we will focus on the left site: the network of C9400-2, C9300-L and C9300-R.

Before getting started, please test the functionality of the network by testing underlay reachability.



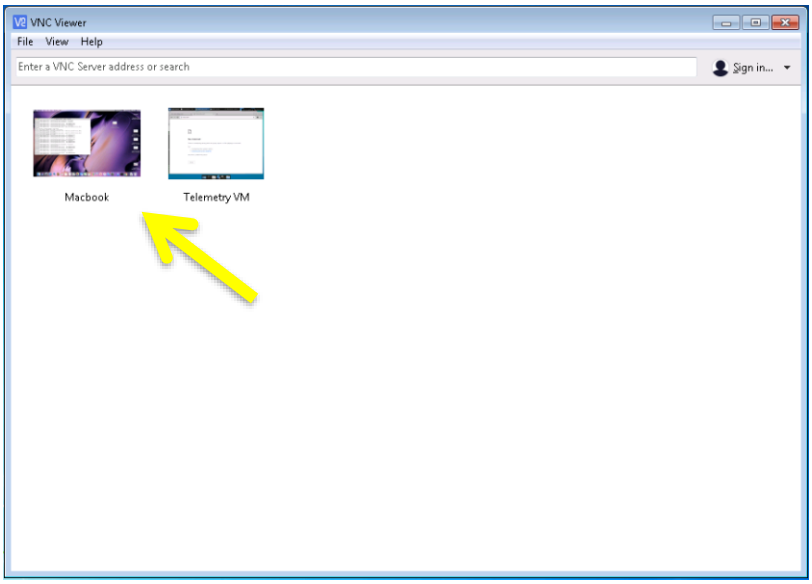
Connect to the C9400-2 via Putty from the taskbar of the jumphost (password if any would be “cisco”).

Check reachability of VLAN 191 (10.11.191.1) from VLAN 190 and 192:

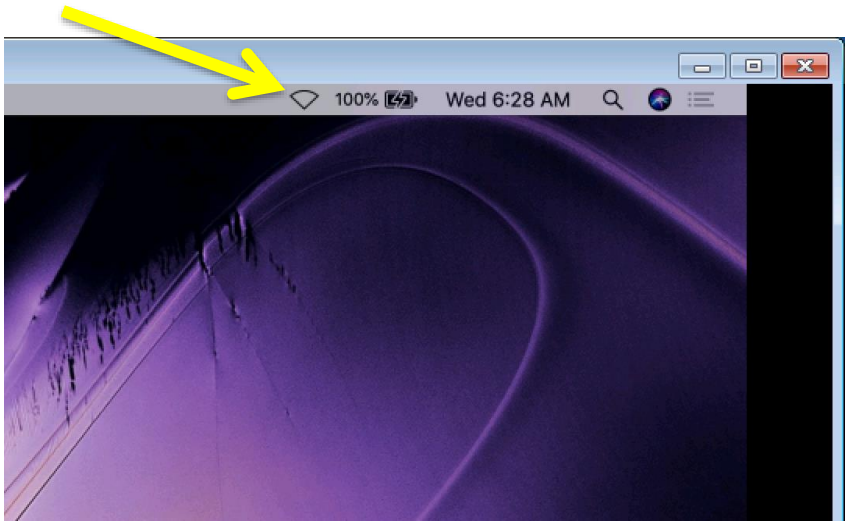
```
C9400-2# ping 10.11.191.1 source vlan 190

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.11.191.1, timeout is 2 seconds:
Packet sent with a source address of 10.11.190.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

C9400-2# ping 10.11.191.1 source vlan 192
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.11.191.1, timeout is 2 seconds:
Packet sent with a source address of 10.11.192.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

Now that we have verified underlay reachability, let us proceed with Bonjour configuration. At this moment, since none of the Bonjour configs have been applied, we shouldn't be able to discover any service such as Apple TV.

Open VNC from taskbar and log into Macbook with password Cisco123!



Look out for AirPlay icon in the Menu Bar on the top right side of the screen. Bonjour services have not yet been configured, which is why the AirPlay icon is not displaying in the menu bar.



Minimize the MacBook VNC window.

Now we shall look at applying Local Area Bonjour policies. The SDG Agent/mDNS gateway resides at the point where an IP gateway exists. In the above topology, even though devices are connected to access switches, their IP gateways reside on 9400-2 and 9300-Data, and so that is where all the Bonjour configs will exist.

First, we need to configure the mdns-sd gateway on our 9400-2 (SDG Agent) which will allow us to configure Bonjour service-lists and definitions:

```
C9400-2(config)# mdns-sd gateway
```

Once configured, we create a list of services that you want to be advertised on the local area. The SDG device only processes whitelisted Bonjour services, while an implicit deny rule at the end of list drops anything not explicitly permitted.

We will configure three ingress Bonjour services:

```
C9400-2(config)#

< Copy Paste in config mode >
       mdns-sd service-definition DEVICE-INFO
       service-type _device-info._tcp.local
       service-type _raop._tcp.local
       mdns-sd service-list LOCAL-AREA-SERVICES-IN IN
       match airplay
       match google-chromecast
       match apple-windows-fileshare
       match DEVICE-INFO
```

Notice how you simply need to type in the service name, and not the .tcp protocol (also known as the service PoinTeR [PTR]) associated with the particular service. IOS takes care of that for you for the case of most commonly used Bonjour services. We can also

create custom service definitions, as we did in the above configuration with *"DEVICE-INFO"*.
Now we will create Egress policies for inter-VLAN communication, which specifies what if any, services from the local area can be advertised across subnets/VLANs:

```
C9400-2(config)#
< Copy Paste in config mode >

      mdns-sd service-list LOCAL-AREA-SERVICES-OUT OUT
        match airplay
        match google-chromecast
        match DEVICE-INFO
```

Now that we've created allowed services, we need to create policy to which we will assign those services-lists. This allows consolidation of many services into one (or more) policies. Do this by doing the following:

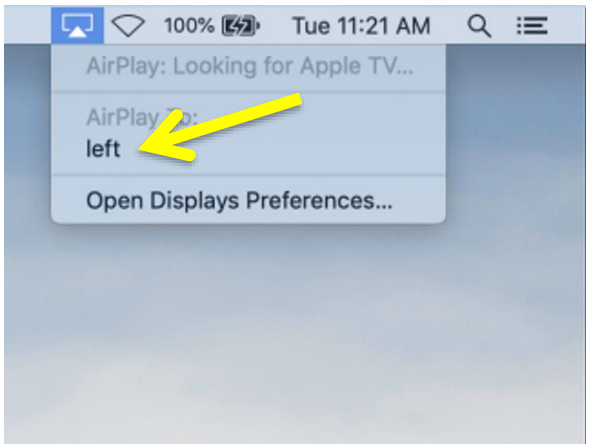
```
C9400-2(config)#
< Copy Paste in config mode >

      mdns-sd service-policy LOCAL-AREA-POLICY
        service-list LOCAL-AREA-SERVICES-IN in
        service-list LOCAL-AREA-SERVICES-OUT out
```

From here, we can now assign these policies to work on specific VLANs. On 9400-2, we have Bonjour endpoints such as Chromecast and Apple TV residing on VLAN 190 and VLAN 192:

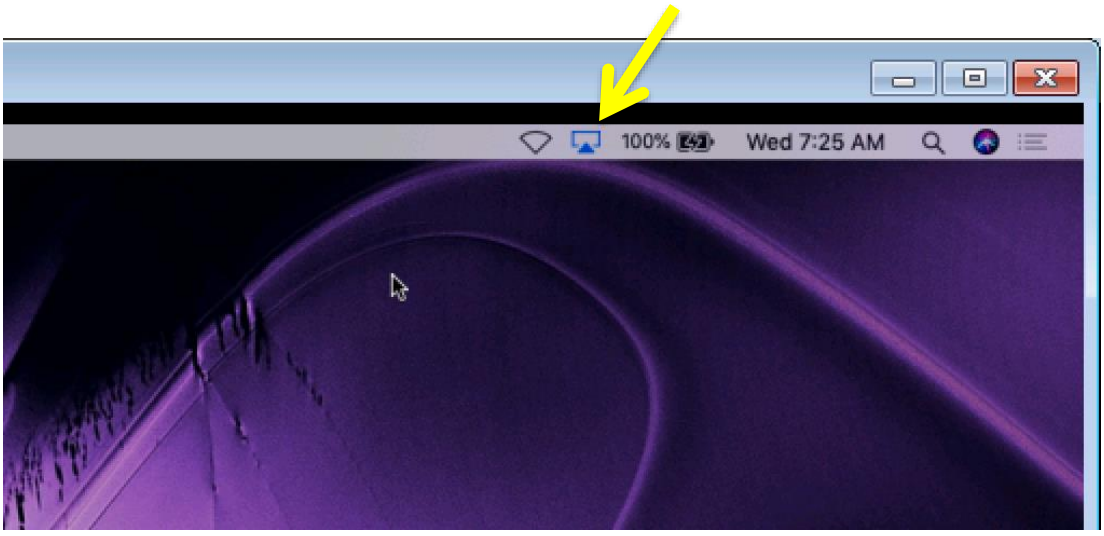
```
C9400-2(config)#
< Copy Paste in config mode >


      interface vlan 190
        mdns-sd gateway
        service-policy LOCAL-AREA-POLICY
        active-query timer 60
      interface vlan 192
        mdns-sd gateway
        service-policy LOCAL-AREA-POLICY
        active-query timer 60
```

The active-query timer command instructs the SDG agent to query for all available services every x seconds, which in our case is 60 seconds.

Now, re-open MacBook VNC and see if you can AirPlay to the AppleTV. Notice how the AirPlay icon is present now, and our AppleTV is showing:



Click on “left” and wait for a few seconds. AirPlay mirroring will begin and the AirPlay icon in the Menu Bar will turn blue:



Now look at all the available services within the network that resides within the 9400-2 SDG Agent.

```
C9400-2# show mdns-sd cache
                    mDNS CACHE
============================================================================================================================
[<NAME>]                    [<TYPE>] [<TTL>/Remaining] [If-name] [Mac Address] [<RR Record Data>]
_googlecast._tcp.local                  PTR   120/88    Vl192  00e0.4c36.d611  Chromecast-11ebe40f52d212fff886269cecc3ac74._googlecast._tcp
Chromecast-11ebe40f52d212fff886269cecc3ac74._googl   SRV   120/88    Vl192  00e0.4c36.d611  0   0   8009   11ebe40f-52d2-12ff-f886-269cecc3ac74.local
11ebe40f-52d2-12ff-f886-269cecc3ac74.local      A     120/88    Vl192  00e0.4c36.d611  10.11.192.4
Chromecast-11ebe40f52d212fff886269cecc3ac74._googl   TXT   4500/4468   Vl192  00e0.4c36.d611
(166)'id=11ebe40f52d212fff886269cecc3ac74"cd=56518789AEFFC7EA85A845FE8BBED9~'~
_airplay._tcp.local                     PTR   4500/4468   Vl192  28ff.3c9d.2c40  left._airplay._tcp.local
_raop._tcp.local                        PTR   4500/4468   Vl192  28ff.3c9d.2c40  28FF3C9D2C40@left._raop._tcp.local
left._airplay._tcp.local                SRV   120/88    Vl192  28ff.3c9d.2c40  0   0   7000   left-24.local                28FF3C9D2C40@left._raop._tcp.local          SRV
120/88    Vl192  28ff.3c9d.2c40  0   0   7000   left-24.local                left-24.local                A     120/110   Vl192  28ff.3c9d.2c40  10.11.192.3
left._airplay._tcp.local                TXT   4500/4468   Vl192  28ff.3c9d.2c40  (342)'acl=0"deviceid=28:FF:3C:9D:2C:40"features=0x5A7FFFF7,0x4155FDE"flags=~'~
28FF3C9D2C40@left._raop._tcp.local      TXT   4500/4468   Vl192  28ff.3c9d.2c40  (192)'cn=0,1,2,3"da=true"et=0,3,5"ft=0x5A7FFFF7,0x4155FDE"sf=0x244"md=0,1,2'~
```

Notice how we can see the name of the .tcp protocols (PTR) running, its associated service (SRV) instance, the A record, and TXT file. If all 4 are present, then the Bonjour service is functional. Furthermore, we get additional information such as the TTL (Time To Live), VLAN, Mac Address, and Record Data. This simplifies the user’s ability to

troubleshoot which services are running, and which are not. It also helps to determine if an issue a network error or device flaw. Notice how we can see an Apple TV as well as Google Chromecast on our cache, which is what we expect based on the service filter that was created.

Step 2: Configure Wide Area Bonjour domain

Wide-Area SDG Domain – The Controller-based solution Wide Area Bonjour domain. The Bonjour gateway role and responsibilities of Cisco Catalyst switching is extended from SDG to an SDG- Agent. The network-wide distributed SDG-Agent devices establish lightweight, stateful and reliable communication channels with a centralized Cisco DNA Center Controller running the Wide Area Bonjour application. The service routing between SDG-Agents and Controller operates over regular IP networks using reliable TCP port 9991 between Cisco DNA Center and SDG-Agent devices. The SDG-Agent must route locally discovered services based on export policy.

The Following sub-section describes the function of each policy type in Local Area domains:

- Wide Area Bonjour – Egress Policy – This mandatory egress policy is applied on a Cisco SDG-Agent device towards Cisco DNA-Center to permit local service routing and discovery request of remote services. The SDG-Agent device only exports whitelisted Bonjour services , while an implicit deny rule at the end of list drops anything not explicitly permitted.

- Wide Area Bonjour – Global Policy – This mandatory bi-directional policy is applied in the Cisco Wide Area Bonjour application of Cisco DNA Center. The single global policy structure is divided into two areas –

    - Source SDG-Agent – The network administrator can assign one or more SDG-Agent announcing selected list of service-types to Cisco DNA Center. Cisco DNA Center accepts Bonjour services from the source only if it matches all selected criteria such as service- type and source network address of the service-provider. For more granular policy configuration, the network address can be IPv4 and IPv6 or alternatively it can be non-specific with Any selection.

    - Receiver SDG-Agent – The same policy as Source SDG-Agent(s), it may consist of one more receiver SDG-Agent(s). Cisco DNA Center accepts and permits Bonjour discovery requests from a receiver only if it matches all selected criteria such service-type and receiver network address of the Bonjour endpoint. For more granular policy configuration, the network address can be IPv4 and IPv6 or alternatively it can be non-specific with Any selection. Based on the service routing topology, Cisco DNA Center

may distribute the services discovered from one or more source SDG-Agent devices to targeted one or more receiver SDG-Agent devices.

Wide Area Bonjour configurations build on top of the Local Area. In order to save you time, we have already configured the right-site(C9300-Data) for you. Therefore, you only need to configure the left-site. Go back into the 9400-2 CLI and configure the Cisco DNA Center controller services as follows:

```
C9400-2(config)#
< Copy Paste in config mode >

      mdns-sd service-list DNAC-CONTROLLER-SERVICES OUT
        match airplay
        match google-chromecast
        match DEVICE-INFO
```

Now that we've created allowed services, we need to create a global policy to which we will assign those services (like we did with Local Area):

```
C9400-2(config)#
< Copy Paste in config mode >

       mdns-sd service-policy DNAC-CONTROLLER-POLICY
         service-list DNAC-CONTROLLER-SERVICES OUT
```

Configure the SDG Controller as 10Gbps Enterprise Port on Cisco DNAC. In this topology that IP address happens to be "10.1.211.40":

```
C9400-2(config)#
< Copy Paste in config mode >

      service-export mdns-sd controller WIDE-AREA-BONJOUR-POLICY
        controller-address 10.1.211.40
        controller-port 9991
        controller-service-policy DNAC-CONTROLLER-POLICY OUT
        controller-source-interface vlan 190
        end
```

Bonjour endpoints inherently advertise and query both on ipv4 as well as ipv6 along with request for PTR, SRV and A records. Since SDG Agents only need to exchange PTR records for the list of available services, we are creating "filters" to only send ipv4 and PTR records from VLAN 190 and VLAN 192.

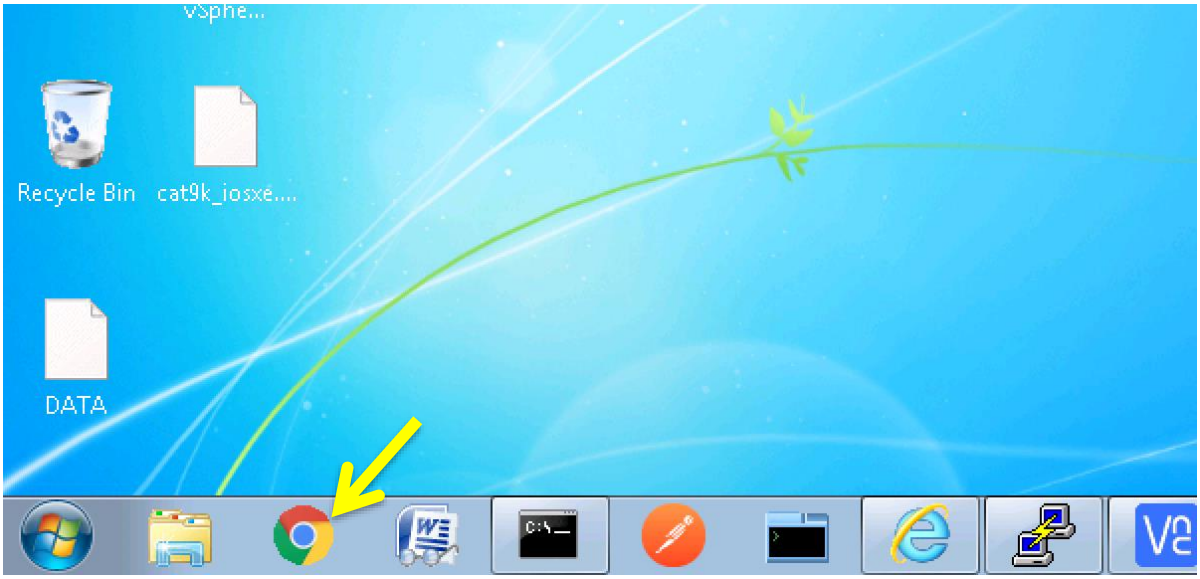
```
C9400-2(config)#
< Copy Paste in config mode >

      int vlan 190
        mdns-sd gateway
        transport ipv4
        service-mdns-query ptr
```

```
int vlan 192
  mdns-sd gateway
  transport ipv4
  service-mdns-query ptr

  end
```

Now check the status of the connection between the SDG Agent and SDG Controller:

```
C9400-2# show mdns-sd controller summary
Controller Summary
====================================
 Controller Name  :   WIDE-AREA-BONJOUR-POLICY
 Controller IP    :   10.1.211.40
 State            :   NEGOTIATING
 Port             :   9991
 Interface        :   Vlan190
 Filter List      :   DNAC-CONTROLLER-POLICY
 Dead Time        :   00:02:00
 Service Buffer   :   Enabled
```

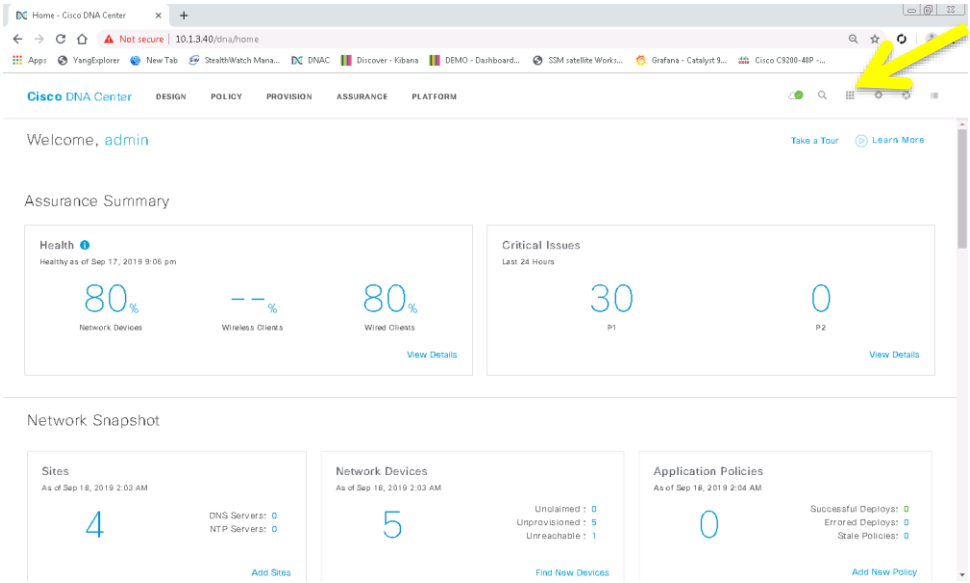
The state is expected to be “Negotiating”, and it will remain so until the Wide Area Bonjour app is configured in Cisco DNA Center.
Open up Chrome by clicking on the icon in the Windows 7 Task Bar:



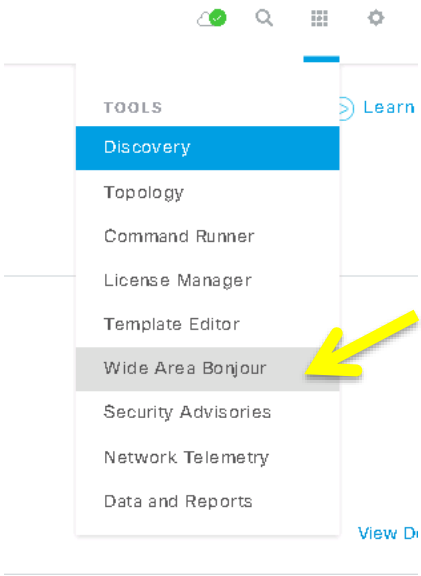
In Google Chrome, click on “DNAC” bookmark (or navigate to the address “10.1.3.40”).
Username: *admin*
Password: *Uabootcamp1*

Click “Log In”

Once logged in, click on the “Tools” icon on top right corner.

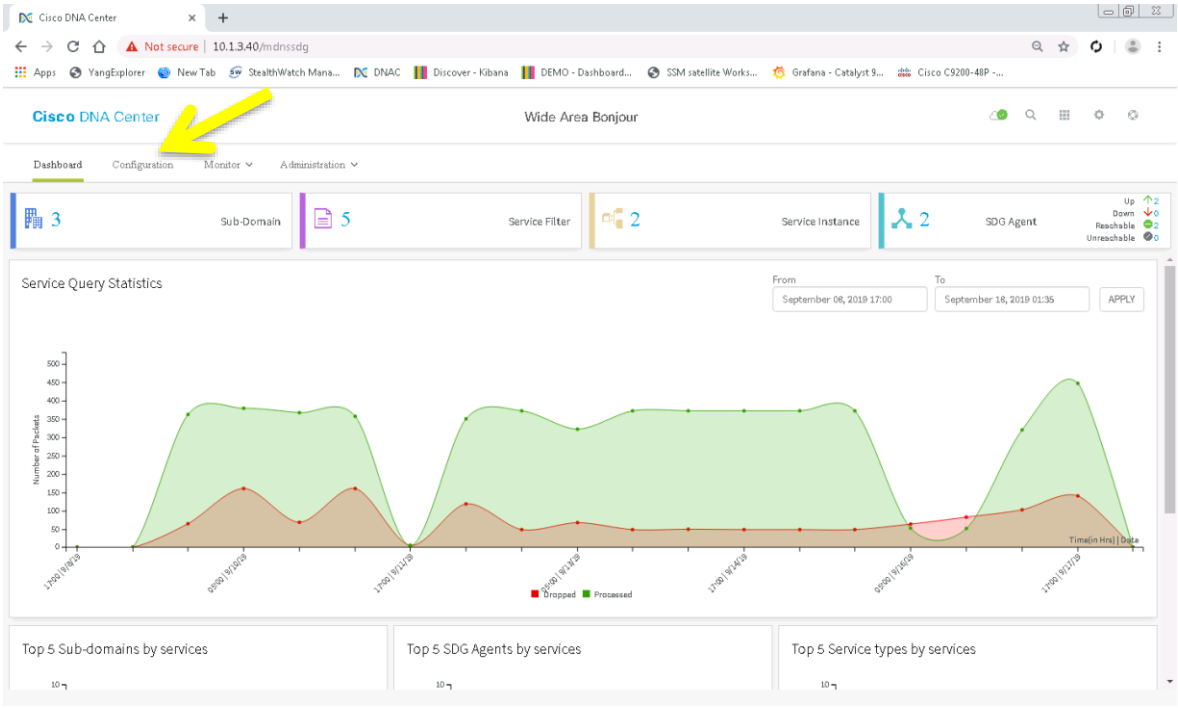


Note: Generally, you would need to install the Wide Area Bonjour app onto Cisco DNA Center, and also run a discovery to discover all the devices on your network. For the purposes of this lab, and lack of time, we have installed the Wide Area Bonjour app and also completed device discovery for you.

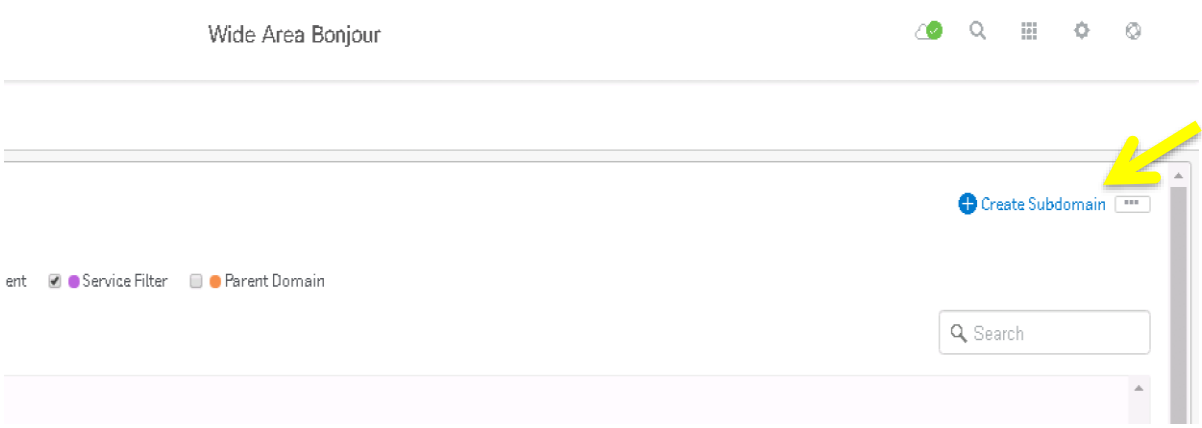


In the “Tools” dropdown, click on the “Wide Area Bonjour”.

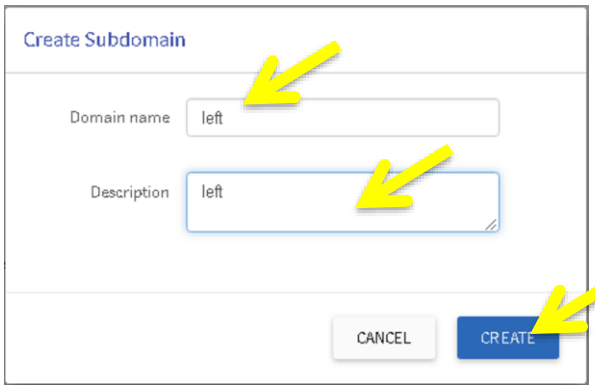
You will be brought into the Wide Area Bonjour app page. Click on “Configuration”:



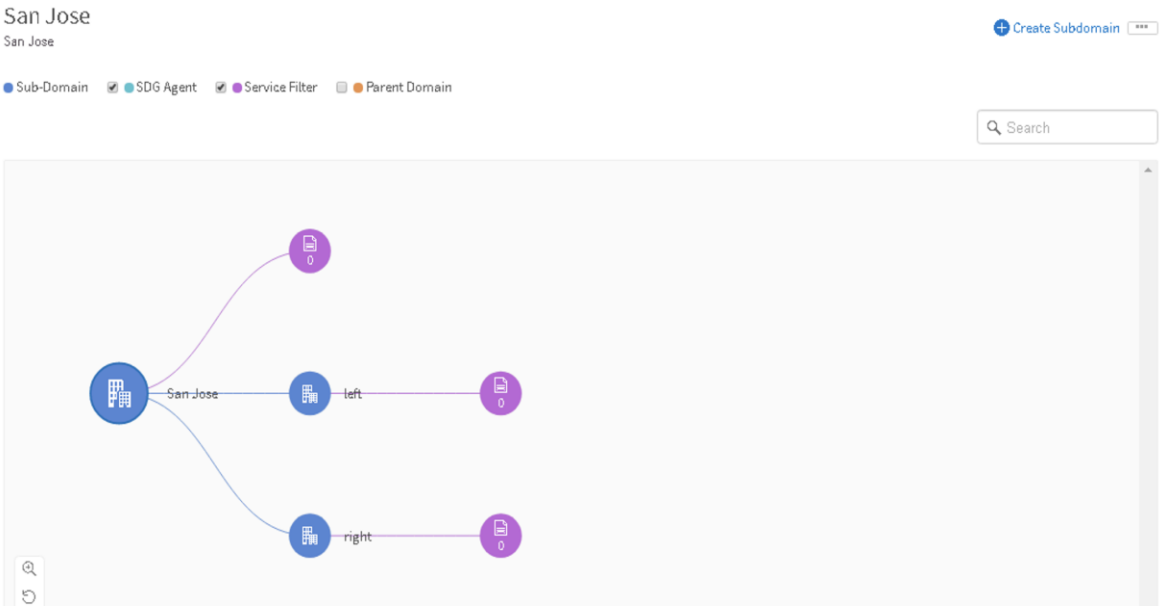
There is a San Jose root domain created for you. You are to create a subdomain “left”. The notion of domain structure and hierarchy in the Cisco Wide Area Bonjour application is to provide network administrators flexible configuration and assurance capabilities to build site and network hierarchies where they would like to build and manage global service-routing policies.



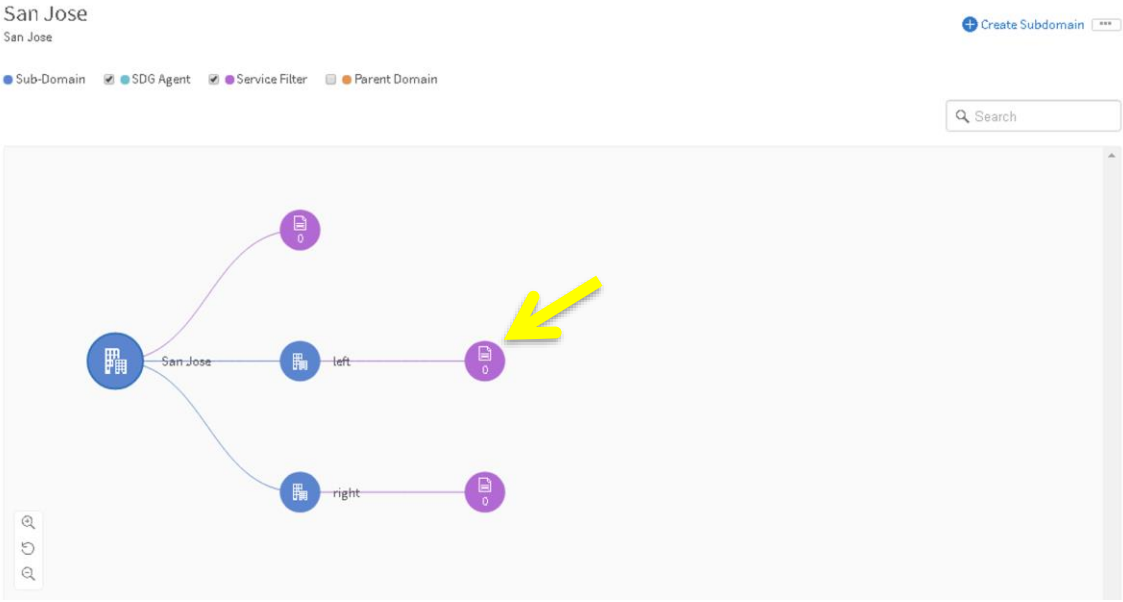
After clicking on the “Create Subdomain” link, fill in the boxes and click “Create”:



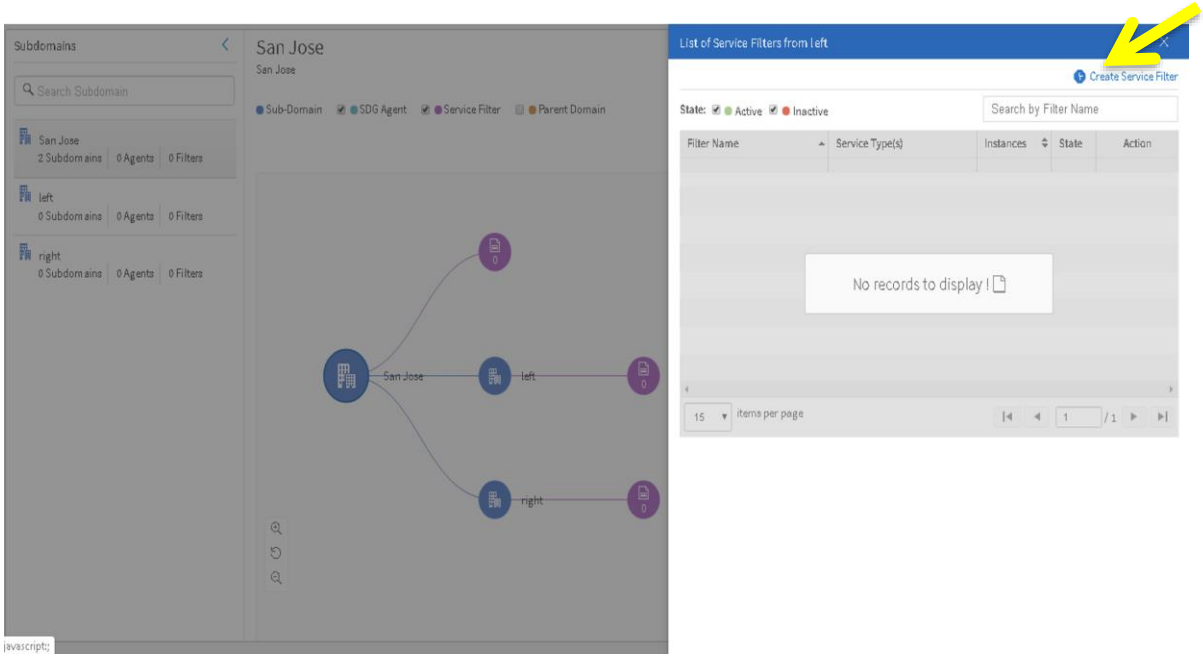
Repeat these steps to create a Subdomain called “right”. At the end, your page should look like this:



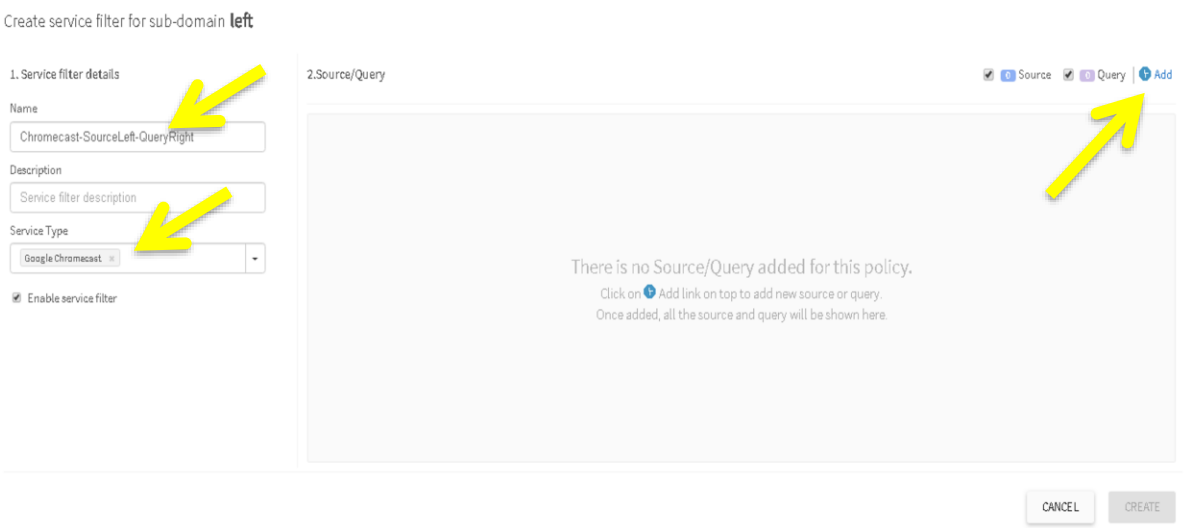
Click on the Service Filter button (pink circles) for subdomain left:



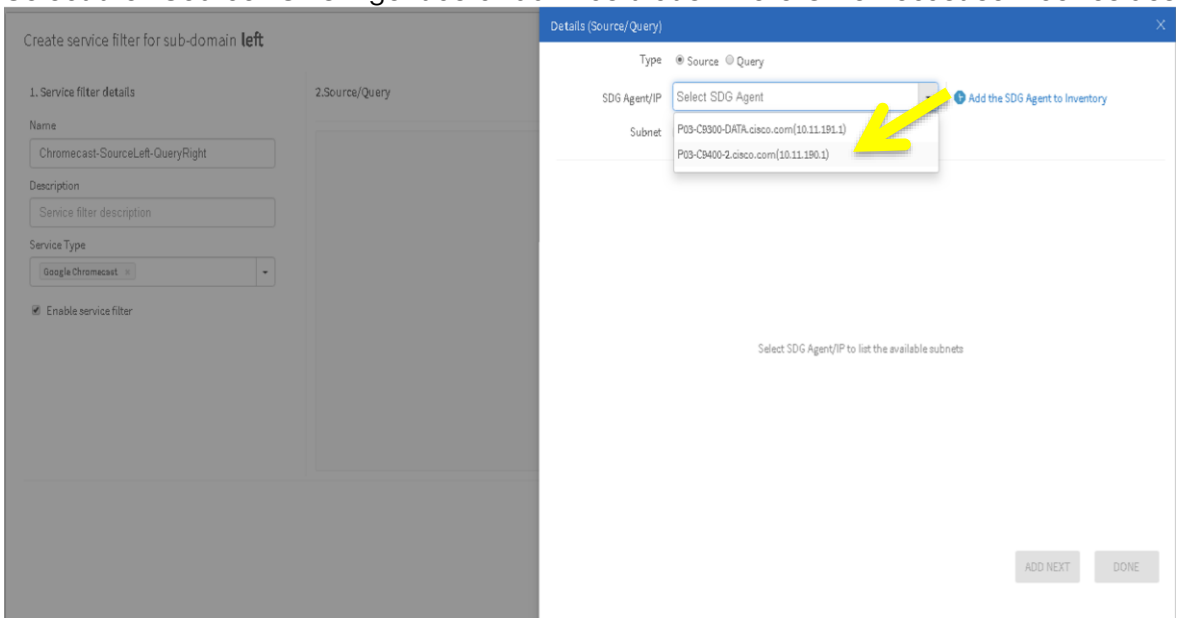
You'll see to the following pop-up screen. On there click on the "Create Service Filter" button



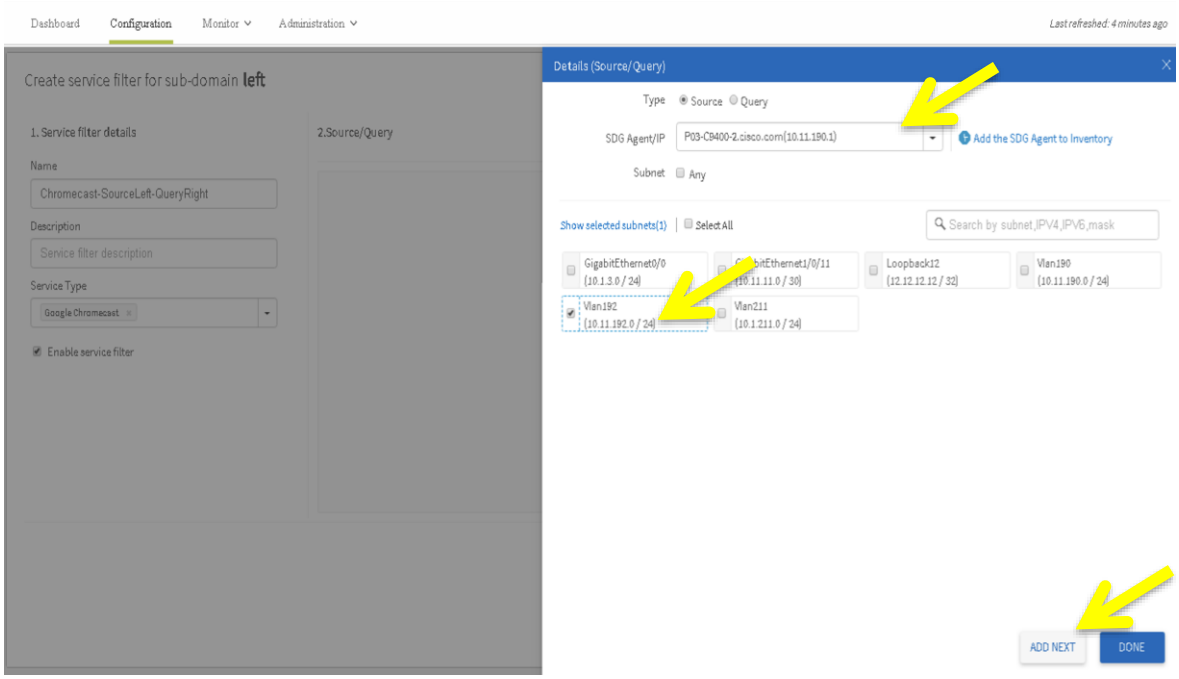
We will now create a policy to advertise Chromecast from the "left" site (i.e. 9400-2) to be seen on the"right" site (i.e 9300-Data). Give the Service Name as "Chromecast-SourceLeft-QueryRight", and from the Service Type drop down select "Google Chromecast" . Then click on the "Add" button to the right side of the window:



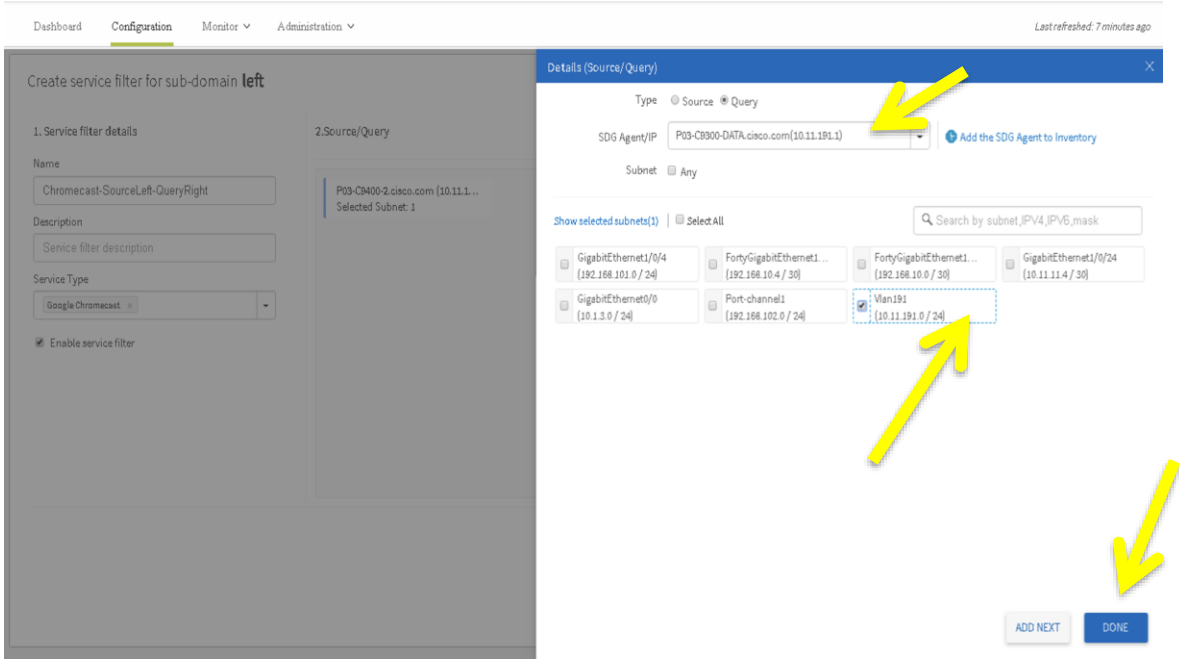
Select the “Source” SDG Agent as 9400-2 as that’s where Chromecast service resides:



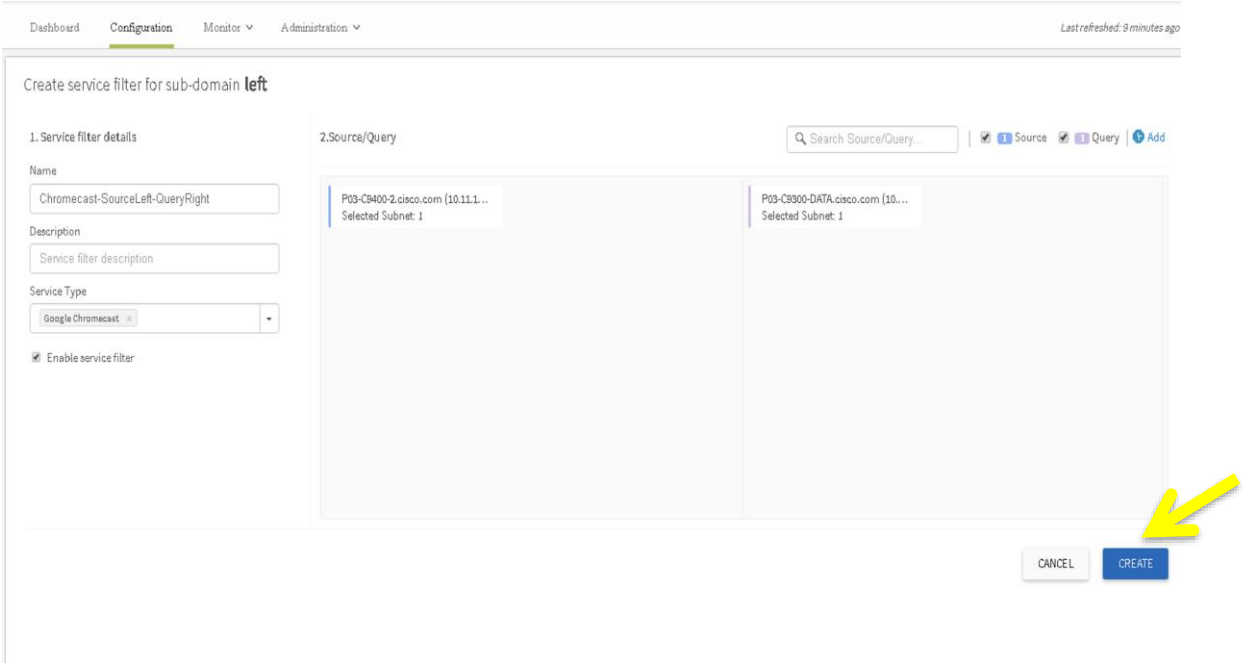
Select the associated subnet as VLAN 192 for 9400-2 and click “Add Next”



Similarly select 9300-Data as Query SDG Agent using subnet VLAN191 and click "Done"



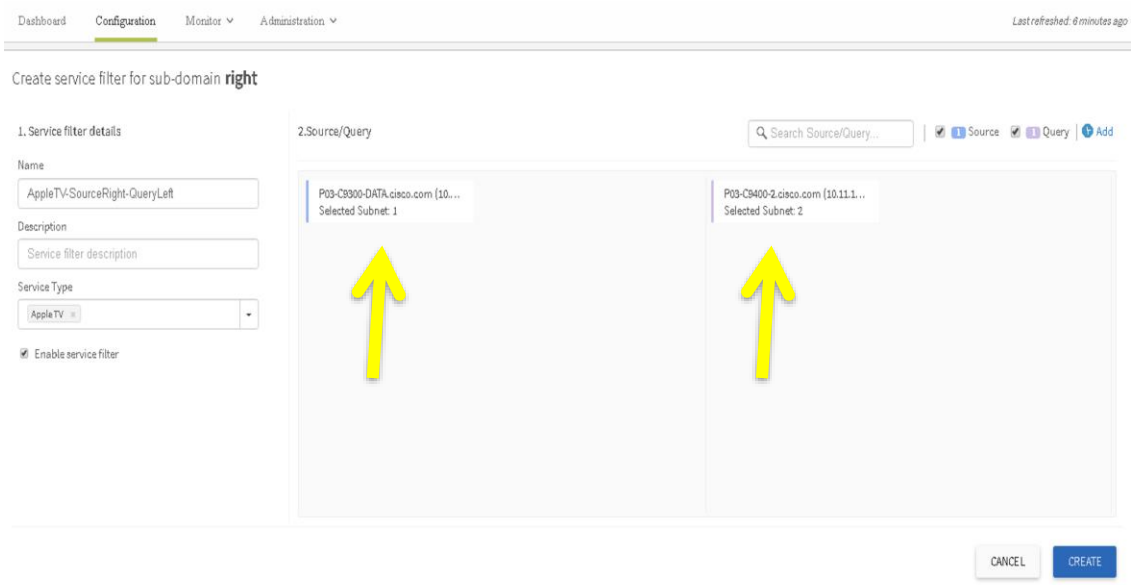
Now click “Create” to enable the service filter.



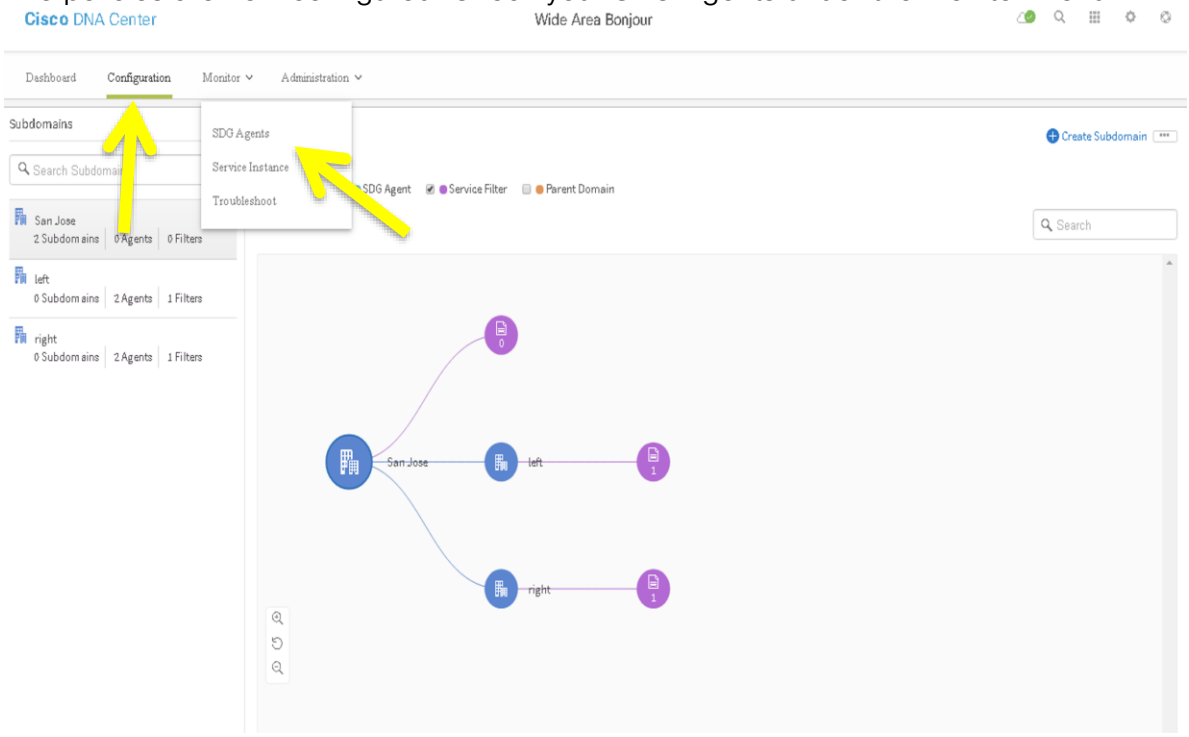
Similarly, we want to advertise Apple TV on the “right” site (9300-Data), so that the Macbook that exists on the “left” site (9400-2) is able to Airplay on it.

To do so, click on the “right” subdomain Service Filter button (pink circle with number 0 in it) and create a service filter “AppleTV-SourceRight-QueryLeft” and select Apple TV from the Service Type drop down menu.

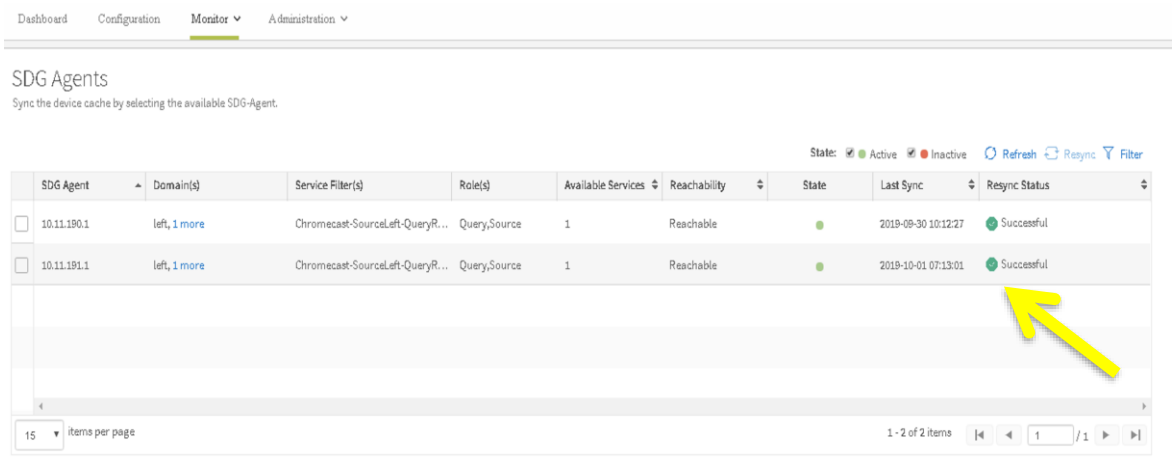
Add C9300-Data as Source SDG Agent on subnet VLAN 191 and 9400-2 as Query SDG Agent on subnet VLAN190 and VLAN192 and it should look something like this:



The policies are now configured. Check your SDG Agents under the Monitor menu:



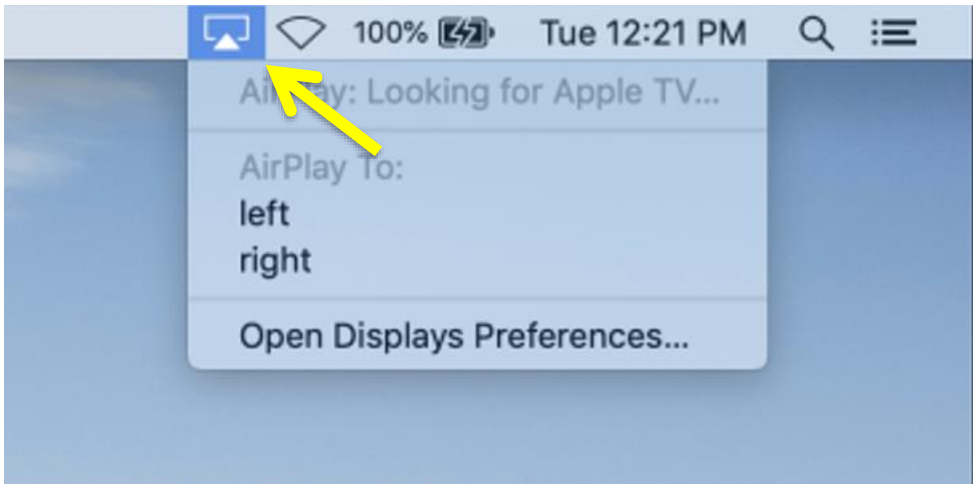
You will now be able to see status of each of the SDG Agents along with the number of services available from each of them.



Now, from the CLI for the C9400-2, run the following command. This should show your state now as “UP”:

```
C9400-2# show mdns-sd controller summary
Controller Summary
=====================================
 Controller Name   :   WIDE-AREA-BONJOUR-POLICY
 Controller IP     :   10.1.211.40
 State             :   UP
 Port              :   9991
 Interface         :   Vlan190
 Filter List       :   DNAC-CONTROLLER-POLICY
 Dead Time         :   00:02:00
 Service Buffer    :   Enabled
```

Now VNC back into the MacBook. You will be able to see two AppleTVs in the Airplay drop-down one “left” from our local area and another one “right” from the Wide Area policy that we configured.



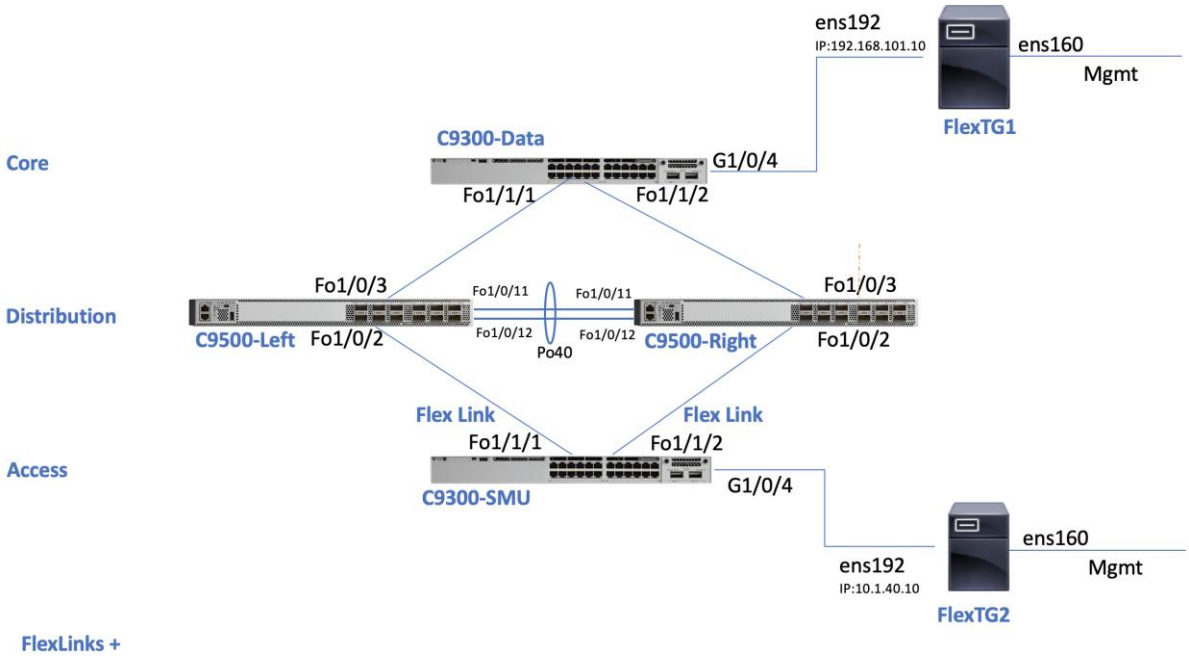
Cisco DNA Service for Bonjour is an enterprise-grade Wide Area Bonjour solution designed to seamlessly integrate into complex wired and wireless network infrastructures. The Cisco Wide Area Bonjour solution retains original end-user experience for using Bonjour technology in complex Enterprise networks. In addition, the new solution provides plug-n-play service-routing capabilities without any forklift changes in DHCP/DNS servers or manual MAC address management. The new distributed architecture supports unparalleled scale, performance, security and redundancy that offers a vendor agnostic compatible solution to enable an end-to-end, services-rich network infrastructure between computers, IoT devices and more.

Scenario Configure Flexlinks +

The Flexlink+ feature enables the user to configure a pair of a Layer 2 interfaces (trunk ports or port channels) where one interface is configured to act as a backup to the other. The feature provides an alternative solution to the Spanning Tree Protocol (STP). Users can disable STP and still retain basic link redundancy. Flexlinks are typically configured in service provider or enterprise networks where customers do not want to run STP on the device. If the device is running STP, Flexlinks are not necessary because STP already provides link-level redundancy or backup.

Flexlink+ can be on the same device or on another device in the stack. When one of the links is up and forwarding traffic, the other link is in standby mode, ready to begin forwarding traffic if the active link shuts down. If the primary link shuts down, the standby link starts forwarding traffic. When the active link comes back up, it goes into standby mode and does not forward traffic. The maximum number of Flexlink+ links that can be configured on a switch or a stack of switches is 26.

Network Diagram FlexLinks +

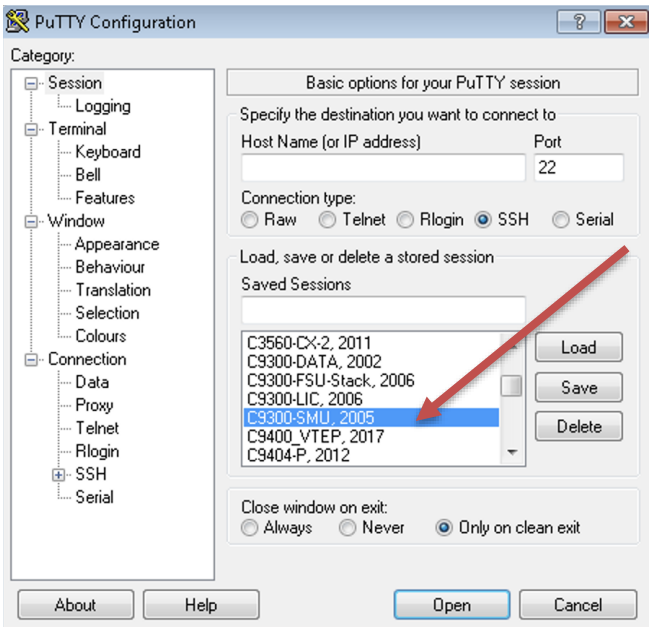


In this lab we will be using the Catalyst 9300 (C9300-SMU) switch as the Device Under Test (DUT). We will be configuring Flexlink+ on C9300-SMU fortyGigabitEthernet uplink interfaces (Fo1/1/1 and Fo1/1/2). It's a typical campus 3 tier architecture with C9500 as a distribution layer and C9300 as a core. There is a L2/L3 demarcation from access to distribution layer. Traffic is generated using iperf3.

Task 1: Configure FlexLinks +

Step 1: Connect to Switches

Connect to C9300-SMU switch using putty and verify the configuration using show commands via CLI



Once connected, check the configuration on interfaces fortyGigabitEthernet 1/1/1 and fortyGigabitEthernet 1/1/2

```
C9300-SMU# show run interface fortyGigabitEthernet 1/1/1
Building configuration...
Current configuration : 43 bytes
interface FortyGigabitEthernet1/1/1
 description ***FLEXLINK+ port to C9500-LEFT***
 switchport trunk allowed vlan 20,40
 switchport mode trunk
end
C9300-SMU# show run interface fortyGigabitEthernet 1/1/2
Building configuration...
Current configuration : 43 bytes
interface FortyGigabitEthernet1/1/2
 description ***FLEXLINK+ port to C9500-RIGHT***
 switchport trunk allowed vlan 20,40
 switchport mode trunk
end
```

Check if there is any REP segment configured on the switch by issuing the *show rep topology* command.

Note: The reason to verify the REP topology is to make sure we do not use the same segment number that has already been configured in the network. If the REP segment is configured it will show the segments that have already been created.

```
C9300-SMU# show rep topology
C9300-SMU#
```

Step 2: Configure Flexlinks +

Configure Flexlink+ on interface fortyGigabitEthernet 1/1/1 and fortyGigabitEthernet 1/1/2

Configuring Flexlink+ Primary/Active Link:

```
C9300-SMU# configure terminal
C9300-SMU(config)# interface fortyGigabitEthernet 1/1/1
C9300-SMU(config-if)# rep segment ?
  <1-1024>  Between 1 and 1024
```

Note: Any number from the range 1-1024 can be used for the segment. Active and Standby links must have the same segment number. A maximum of 26 Flexlink+ segments can be configured on the supported platforms. For this lab, we will be creating one Flexlink+ segment.

```
C9300-SMU(config-if)# rep segment 1023 edge no-neighbor primary

Warning: Enabling REP automatically disables STP on this port.  It is recommended to shutdown
all interfaces which are not currently in use to prevent potential bridging loops.
```

Verify the configuration on the interface.

```
C9300-SMU# show run interface fortyGigabitEthernet 1/1/1
Building configuration...
Current configuration : 194 bytes
!
interface FortyGigabitEthernet1/1/1
 description ***FLEXLINK+ port to C9500-LEFT***
 switchport trunk allowed vlan 20,40
 switchport mode trunk
 rep segment 1023 edge no-neighbor primary
end
```

Configuring Flexlink+ Secondary/Standby Link:

```
C9300-SMU# configure terminal
C9300-SMU(config)# interface fortyGigabitEthernet 1/1/2
C9300-SMU(config-if)# rep segment 1023 edge no-neighbor
```

Verify the configuration on the interface.

```
C9300-SMU# show run interface fortyGigabitEthernet 1/1/2
Building configuration...

Current configuration : 187 bytes
!
interface FortyGigabitEthernet1/1/2
 description ***FLEXLINK+ port to C9500-RIGHT***
 switchport trunk allowed vlan 20,40
 switchport mode trunk
 rep segment 1023 edge no-neighbor
end
```

Verify if the Flexlink+ has been configured on the device by issuing the “Show rep topology” command.

```
C9300-SMU# show rep topology
REP Segment 1023
BridgeName                       PortName    Edge Role
-------------------------------- ----------- ---- ----
C9300-SMU                        Fo1/1/1     Pri* Open
C9300-SMU                        Fo1/1/2     Sec* Alt
```

We can see that Flexlink+ with REP segment 1023 has been configured on the device. Interface FortyGigabitEthernet1/1/1 is configured as the primary link and is open to pass traffic. Interface FortyGigabitEthernet1/1/2 is the secondary interface and alternate port for the Flexlink+ segment. In case the Primary/active port goes down, the secondary port will start passing traffic.

Next, We nee to stop the TCNs from link going down on the Distribution switches.

NOTE: If TCN is not disabled the Distribution switch, STP reconvergence will take longer regardless how fast the FLEX link converges we will see drops during that time. Hence we need to disable TCNs from these ports.

```
C9500-LEFT# configure terminal
C9500-LEFT(config)# interface fortyGigabitEthernet 1/0/2
C9500-LEFT(config-if)# spanning-tree portfast trunk
C9500-LEFT(config-if)# spanning-tree bpduguard enable

C9500-RIGHT# configure terminal
C9500-RIGHT(config)# interface fortyGigabitEthernet 1/0/2
C9500-RIGHT(config-if)# spanning-tree portfast trunk
C9500-RIGHT(config-if)# spanning-tree bpduguard enable
```

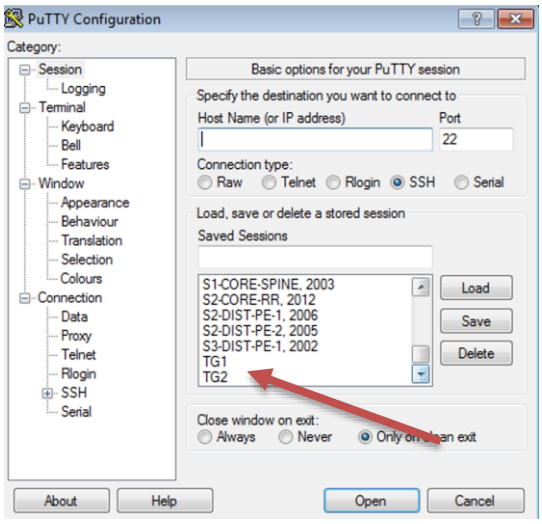
< Copy Paste in config mode >

```
configure terminal
   interface fortyGigabitEthernet 1/0/2
      spanning-tree portfast trunk
      spanning-tree bpduguard enable
end
```

Step 3: Run traffic

Start the traffic with ping between two End hosts. Open two putty sessions already saved “TG1” and “TG2”

Login: *user*
Password: *nbv_12345*



On "*TG1*", execute the command *ping 10.1.40.10* and keep the ping running.

```
user@FlexTG1:~$ ping 10.1.40.10
PING 10.1.40.30 (10.1.40.10) 56(84) bytes of data.
64 bytes from 10.1.40.10: icmp_seq=1 ttl=253 time=0.792 ms
64 bytes from 10.1.40.10: icmp_seq=2 ttl=253 time=0.983 ms
```

On "TG2" verify the destination IP is hosted on TG2 VM

```
user@FlexTG2:~$ ifconfig
ens192    Link encap:Ethernet  HWaddr 00:0c:29:ac:d9:87
          inet addr:10.1.40.10  Bcast:10.1.40.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feac:d987/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6960115 errors:0 dropped:427 overruns:0 frame:0
          TX packets:390066 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9257145051 (9.2 GB)  TX bytes:25809929 (25.8 MB)
user@FlexTG2:~$

user@FlexTG2:/etc/network$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.1.3.252      0.0.0.0         UG    0      0        0 ens160
10.1.3.0        *               255.255.255.0   U     0      0        0 ens160
10.1.40.0       *               255.255.255.0   U     0      0        0 ens192
192.168.101.0   10.1.40.30      255.255.255.0   UG    0      0        0 ens192
192.168.122.0   *               255.255.255.0   U     0      0        0 virbr0
user@FlexTG2:/etc/network$
```

Ping on the other machine will start again.

You can use the ping and see 0-1 ping/seconds drops on failover

```
user@FlexTG1:~$ ping 10.1.40.10
>>> Keep it running
```

-OR-

You can use IPERF3 which pumps 1 Gbps stream.

On "TG2"

```
user@FlexTG2:/etc/network$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

On "TG1"

```
user@FlexTG1:~$ iperf3 -c 10.1.40.10 -t 5000
Connecting to host 10.1.40.10, port 5201
[  4] local 192.168.101.10 port 35938 connected to 10.1.40.10 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   113 MBytes   947 Mbits/sec    0    458 KBytes
[  4]   1.00-2.00   sec   112 MBytes   944 Mbits/sec    0    482 KBytes
[  4]   2.00-3.00   sec   112 MBytes   943 Mbits/sec    0    482 KBytes
[  4]   3.00-4.00   sec   112 MBytes   938 Mbits/sec    0    482 Kbytes
```

And monitor the performance.

Task 2: Test Time to failover

Step 1: Flip links

Shutdown the Flexlink+ active interface and observe the traffic. Keep both the switch and TG1 windows open so that we can see if there are packet drops. The Flexlink+ convergence time for unicast traffic is <100ms and for multicast traffic it is <50ms. We would see no drops or minimal drops due to this sub-second convergence.

On the C9300-SMU switch, execute the *shutdown* command for interface fortyGigabitEthernet 1/1/1, which is the Primary Flexlink+ interface. After that, execute the *do show rep topology* command to see the state of the Flexlink+ interfaces.

```
C9300-SMU# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9300-SMU(config)# interface fortyGigabitEthernet 1/1/1
C9300-SMU(config-if)# shutdown
C9300-SMU# show  rep topology
REP Segment 1023
Warning: REP detects a segment failure, topology may be incomplete
BridgeName                       PortName   Edge Role
-------------------------------- ---------- ---- ----
C9300-SMU                        Fo1/1/2    Sec* Open
C9300-SMU                        Fo1/1/1    Sec* Fail
C9300-SMU#
```

We can observe that there are no, or minimal, traffic drops. When we check the rep topology, Fo1/1/1 is now showing as Fail, while Fo1/1/2 is in Open state and started forwarding the traffic.

Now, execute *no shutdown* on Fo1/1/1, and check the rep topology again.

```
C9300-SMU# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9300-SMU(config)# int fo1/1/1
C9300-SMU(config-if)# no shut
C9300-SMU# show  rep topology
REP Segment 1023
BridgeName                       PortName   Edge Role
-------------------------------- ---------- ---- ----
C9300-SMU                        Fo1/1/1    Pri* Alt
C9300-SMU                        Fo1/1/2    Sec* Open
```

We can see that the interface roles have been changed and interface Fo1/1/1, the previous Primary interface, is now in Alternative state and ready to forward the traffic in case the Open interface, which in our case is Fo1/1/2, goes down.
Now go to interface fortyGigabitEthernet 1/1/2, shutdown the interface and check the rep topology again.

```
C9300-SMU# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9300-SMU(config)# interface  fortyGigabitEthernet 1/1/2
C9300-SMU(config-if)# shutdown
C9300-SMU(config-if)# end
C9300-SMU# show rep topology
REP Segment 1023
Warning: REP detects a segment failure, topology may be incomplete
BridgeName                       PortName   Edge Role
-------------------------------- ---------- ---- ----
C9300-SMU                    Fo1/1/1    Sec* Open
C9300-SMU                    Fo1/1/2    Sec* Fail
```

We can see that interface fortyGigabitEthernet 1/1/2 is in failed state and fortyGigabitEthernet 1/1/1 is open and forwarding the traffic.

```
C9300-SMU# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9300-SMU(config)# interface  fortyGigabitEthernet 1/1/2
C9300-SMU(config-if)# no shutdown
C9300-SMU(config-if)# end

C9300-SMU# show rep topology
REP Segment 1023
BridgeName                       PortName   Edge Role
-------------------------------- ---------- ---- ----
C9300-SMU                        Fo1/1/1    Pri* Open
C9300-SMU                        Fo1/1/2    Sec* Alt

C9300-SMU#
```

Finally, we are back to the original state, and you can see how fast the convergence was.

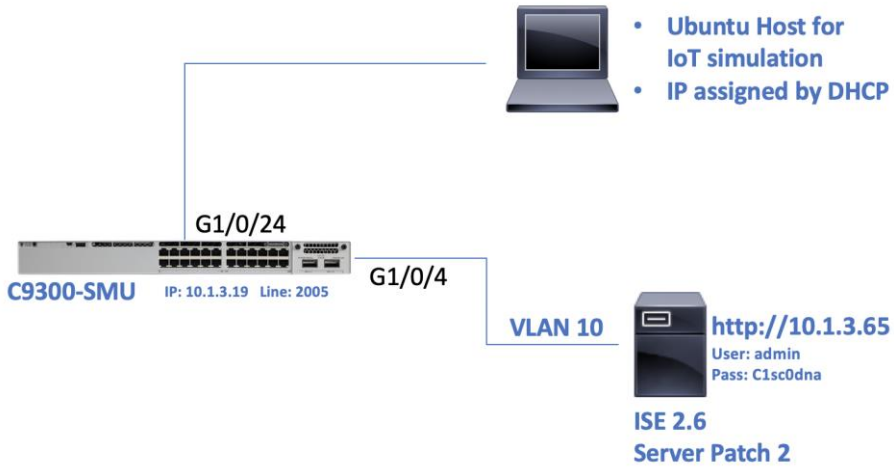
Scenario Configure MUD Services

In this lab activity, you will learn how to auto classify IoT devices like Molex LED lights or other devices which are Manufacturer Usage Description (MUD) protocol capable and then apply Rules based on the classification. In the IOS XE16.9.1 version of software for the Cisco Catalyst 9000 Series of switches, a new “device sensor” feature was added that allows the switches to classify devices based on LLDP TLV 127 and to send that classification to RADIUS via accounting messages.

Manufacturer Usage Description (MUD) is an IETF standard that defines a mechanism for signaling device classification and automatically enforcing “intended” behavior defined by manufacturers in the Enterprise network.

This lab will show you the key configurations and checks to verify how MUD is implemented on Cisco Catalyst 9000 Series switches with ISE 2.6.

Network Diagram MUD

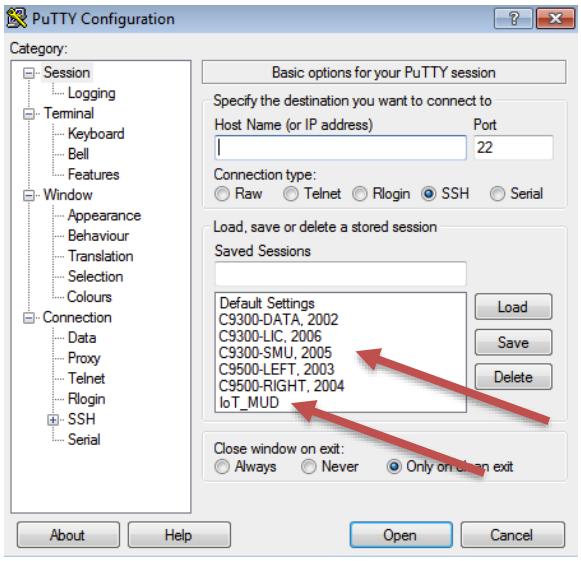


Task 1: Initialize MUD Config

Step 1: Connect to Cat 9000 switch and MUD end host

Connect to C9300-SMU and IoT Host “MUD” via terminal in Putty

IoT MUD Username: mud
IoT MUD Password: mud123



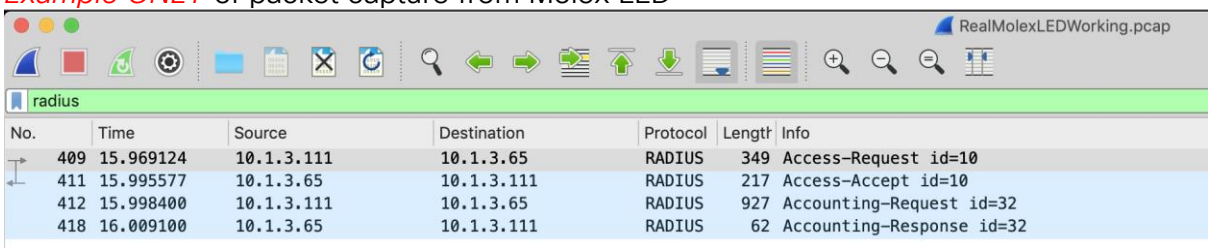
Step 2: Configure Catalyst 9000 switch for MUD

Set MAB on device via RADIUS Server.

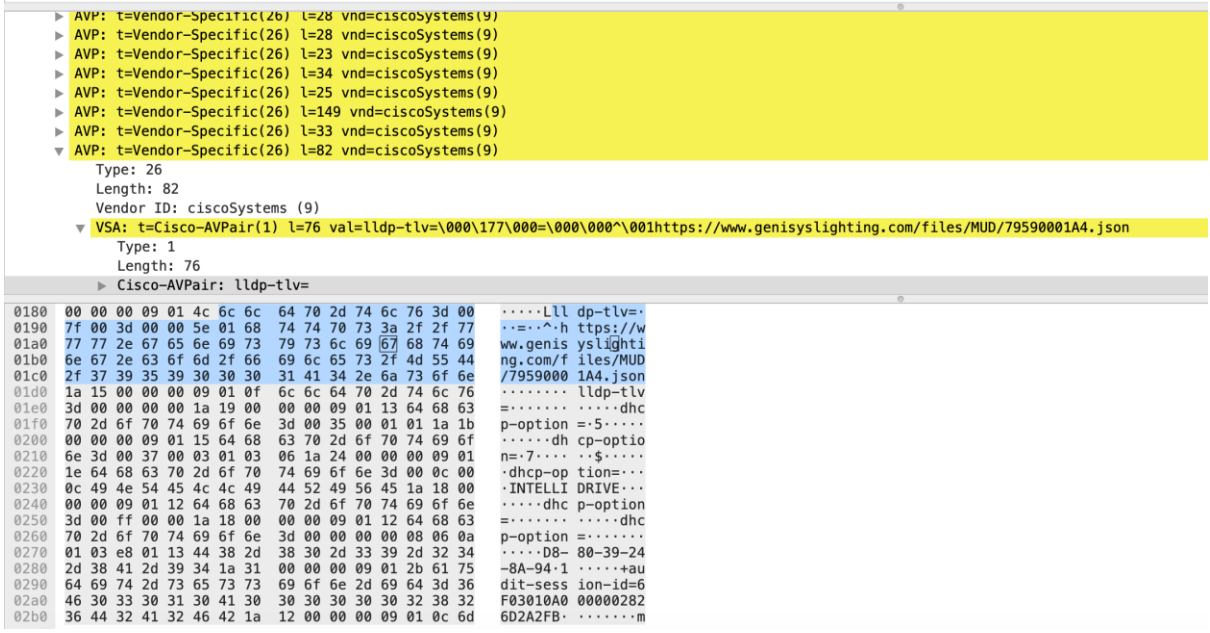
```
C9300-SMU# configure terminal

< Copy Paste in config mode >
aaa new-model

!Ignore Authentication on Console as backup path
aaa authentication login Console none

!Enable dot1x/MAB framework
aaa authentication dot1x default group radius
aaa authorization network default group radius
aaa accounting update newinfo

!Enter This CLI individually to enable IBNS 2.0
aaa accounting identity default start-stop group radius
This operation will permanently convert all relevant authentication
commands to their CPL control-policy equivalents. As this conversion is
irreversible and will disable the conversion CLI 'authentication display
[legacy|new-style]', you are strongly advised to back up your current
configuration before proceeding.
Do you wish to continue? [yes]: yes

!Enable RADIUS accounting used to provide MUDURL
aaa accounting network default start-stop group radius
```

AAA accounting will be used by Device Sensor to provide information from LLDP or DHCP message to RADIUS

Example ONLY of packet capture from Molex LED



Look at the Accounting Request message:



The packet contains the MUDURL attribute which is extracted by the LLDP TLV or DHCP option 161. There are specific URL requirements like:

https://<url>/<file>

For your reference: To Validate the URL is valid use GitHub tool. The Tool uses .pcap file which can be collected from ISE via TCP Dump (discussed later into the guide)

https://github.com/CiscoDevNet/MUD-URL-Validator

```
C9300-SMU(config)#

< Copy Paste in config mode >
! Enable Change of Authorization from RADIUS
aaa server radius dynamic-author
 client 10.1.3.65 server-key cisco123
 server-key cisco123

! Exclude DHCP address if Local DHCP server is used
ip dhcp excluded-address 192.168.20.1 192.168.20.230
ip dhcp excluded-address 192.168.20.250 192.168.20.255
```

```
! Set Local DHCP server
ip dhcp pool MUD
 network 192.168.20.0 255.255.255.0
 default-router 192.168.20.1

! Enable DHCP snooping to be used by Device Sensor to track DHCP requests
and LLDP TLVs
ip dhcp snooping vlan 10
ip dhcp snooping

! Enable Device Sensor to track based on LLDP or DHCP. Device sensor
config is based on IBNS 2.0
device-sensor notify all-changes
access-session attributes filter-list list mudtest
 lldp
 dhcp
access-session accounting attributes filter-spec include list mudtest
access-session monitor

! Enable the LLDP process so the switch will form LLDP neighbours with IoT
devices
lldp run

! Configure a very basic IBNS 2.0 policy to trigger MAB on the interface.
policy-map type control subscriber mud-mab-test
 event session-started match-all
  10 class always do-until-failure
   10 authenticate using mab

! Configure an interface template to scale the config changes
! STP Portfast is a requirement as the MAB can timeout before the port STP
state changes to Forward.
template mud-mab-test
 spanning-tree portfast
 switchport access vlan 10
 switchport mode access
 mab
 access-session port-control auto
 service-policy type control subscriber mud-mab-test

! Configure the interface(s) to use the template
! The interface is kept down to ensure the device does not come up yet
interface GigabitEthernet1/0/24
 source template mud-mab-test
 dot1x pae authenticator
 shutdown
vlan 10

! Set the SVI for NAS to establish communication to RADIUS
interface Vlan10
 ip address 192.168.20.1 255.255.255.0

! Set the RADIUS server preshared key
radius server AAA
 address ipv4 10.1.3.65 auth-port 1645 acct-port 1646
 key cisco123
```

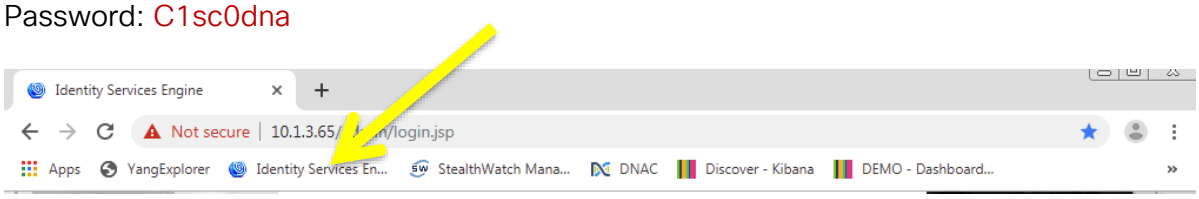
```
! Set the Console session to Exclude AAA authentication as back up
line con 0
 login authentication Console
```

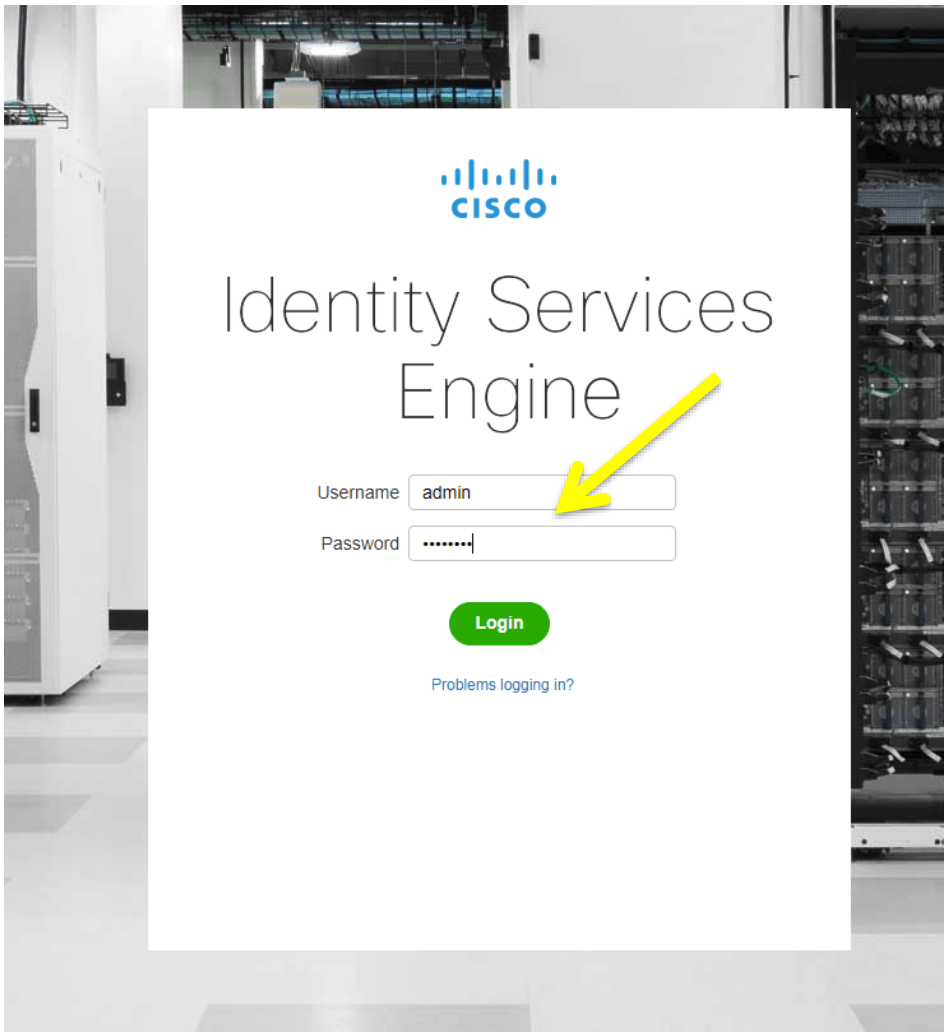
Step 2: Configure ISE to classify based on MUD

Login to ISE via Google Chrome:

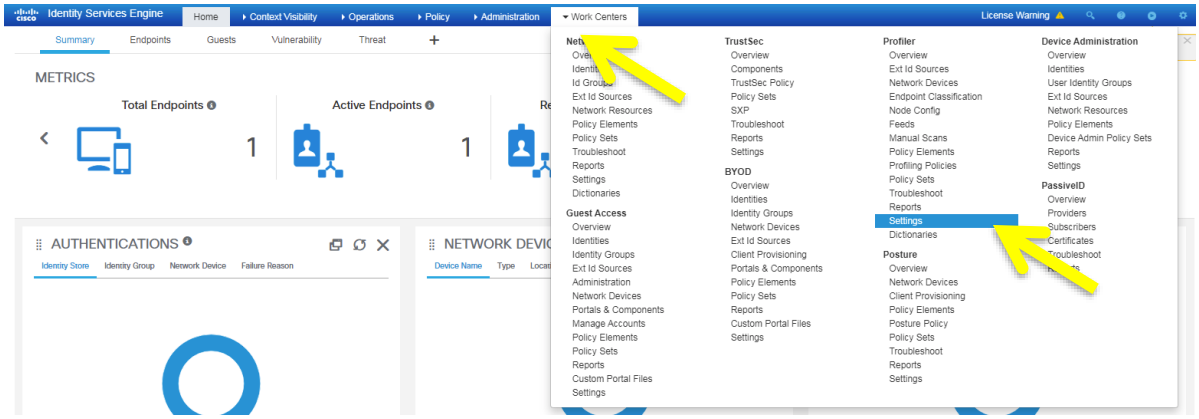
IP: 10.1.3.65

Username: admin
Password: C1sc0dna

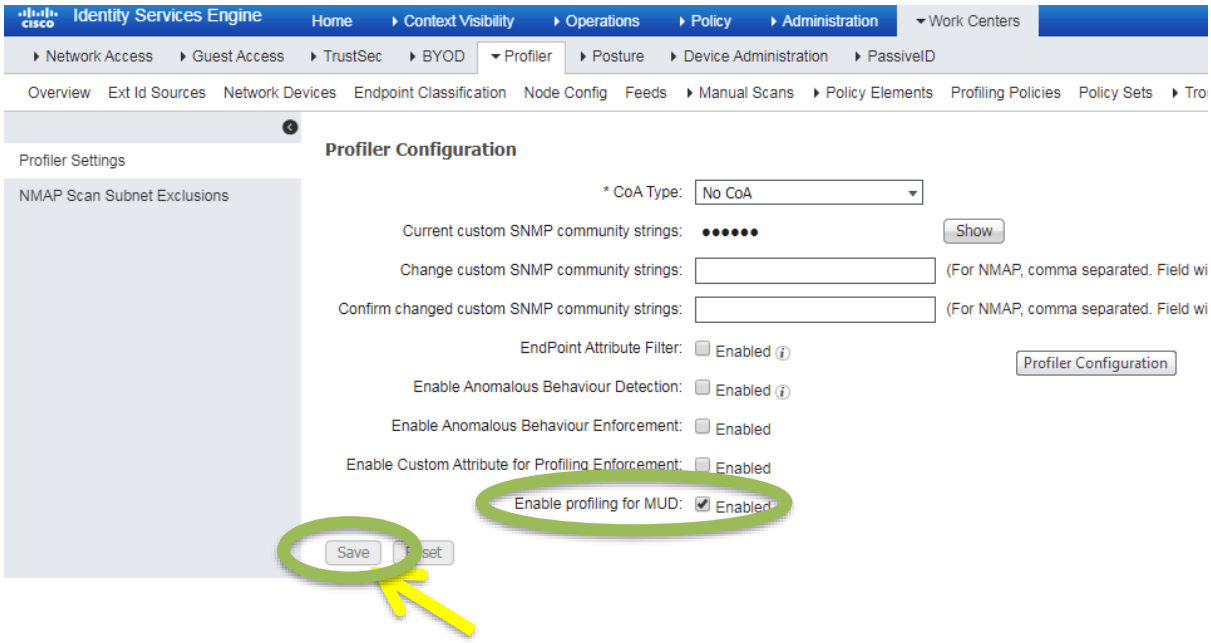




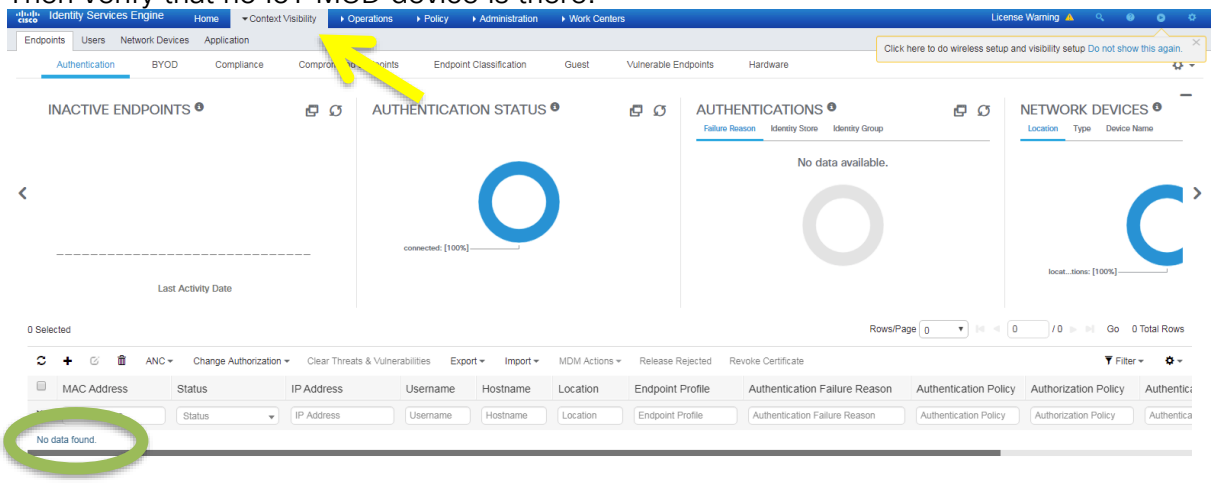
Open ISE Profiler settings:



Enable MUD profiling and Click on “Save”. “Save” button will change to dark blue to indicate that there are changes.



Then verify that no IoT MUD device is there.



Task 2: Start LLDP process on the Ubuntu IoT simulator Engine

Step 1: Go to IoT_MUD host on putty

```
login as: mud
mud@10.1.3.110's password:mud123
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic i686)
```

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Tue Sep 17 11:01:21 2019 from 10.1.3.254
mud@MUD:~$

mud@MUD:~$ sudo lldpd
[sudo] password for mud: mud123 ← if asked
mud@MUD:~$ ps -ef | grep lldp
root     12718     1  0 16:28 ?        00:00:00 lldpd
root     12720 12718  0 16:28 ?        00:00:00 lldpd
mud      12722 12697  0 16:28 pts/1    00:00:00 grep --color=auto lldp
mud@MUD:~$
Enter Custom TLV that simulates the IoT device. URL rules were discussed
above.
mud@MUD:~$ sudo sh lldpmud https://mud.com/pi4.json
[sudo] password for mud: mud123 ← if asked
mud@MUD:~$
```

Step 2: Check if device-sensor has detected the custom TLV

```
! No shutdown interface G1/0/24 and wait for 10-15 seconds for LLDP to come up
C9300-SMU(config)# interface GigabitEthernet1/0/24
C9300-SMU(config-if)# no shutdown
C9300-SMU(config-if)# end
C9300-SMU# show lldp neighbors g1/0/24
Capability codes:
  (R) Router, (B) Bridge, (T) Telephone, (C) DOCSIS Cable Device
  (W) WLAN Access Point, (P) Repeater, (S) Station, (O) Other

Device ID           Local Intf     Hold-time  Capability      Port ID
MUD.cisco.com       Gi1/0/24       120        B,R             a036.9f77.a1d9
Total entries displayed: 1
! Verify that at least 1 neighbor is shown
C9300-SMU# show device-sensor cache interface g1/0/24
Device: a036.9f77.a249 on port GigabitEthernet1/0/24
-------------------------------------------------------------------------------
Proto Type:Name                           Len Value                           Text
LLDP   127:organizationally-specific      30 FE 1C 00 00 5E 01 68 74 74  ....^.htt
                                             70 73 3A 2F 2F 6D 75 64 2E  ps://mud.
                                             63 6F 6D 2F 70 69 34 2E 6A  com/pi4.j
                                             73 6F 6E                    son
LLDP      4:port-description               8 08 06 65 6E 73 31 39 32      ..ens192
LLDP      8:management-address            26 10 18 11 02 FE 80 00 00 00  .....^@...
                                             00 00 00 02 0C 29 FF FE 0E  .....)...
                                             A8 C9 02 00 00 00 02 00     (.......
LLDP      7:system-capabilities            6 0E 04 00 9C 00 14           ...^\..
LLDP      6:system-description            94 0C 5C 55 62 75 6E 74 75 20  .\Ubuntu
                                             31 36 2E 30 34 2E 36 20 4C  16.04.6 L
                                             54 53 20 4C 69 6E 75 78 20  TS Linux
                                             34 2E 34 2E 30 2D 31 34 32  4.4.0-142
                                             2D 67 65 6E 65 72 69 63 20  -generic
                                             23 31 36 38 2D 55 62 75 6E  #168-Ubun
                                             74 75 20 53 4D 50 20 57 65  tu SMP We
                                             64 20 4A 61 6E 20 31 36 20  d Jan 16
                                             32 31 3A 30 31 3A 31 35 20  21:01:15
                                             55 54 43 20 32 30 31 39 20  UTC 2019
                                             69 36 38 36                 i686
LLDP      5:system-name                    5 0A 03 4D 55 44              ..MUD
LLDP      0:end-of-lldpdu                  2 00 00                       ..
LLDP      3:time-to-live                   4 06 02 00 78                 ...x
LLDP      2:port-id                        9 04 07 03 A0 36 9F 77 A2 49  ....6.w"I
LLDP      1:chassis-id                     9 02 07 04 00 0C 29 0E A8 C9  .....).(.
C9300-SMU#
```

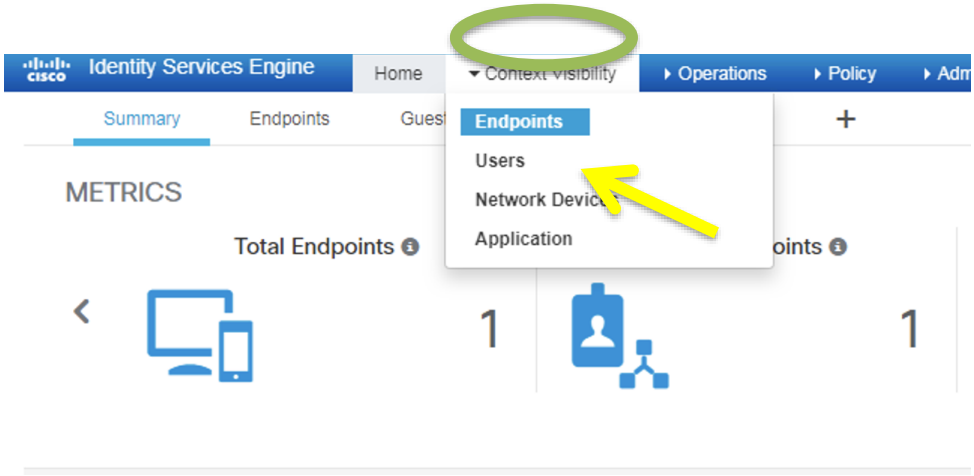
```
! the MUD specific attribute is Type 127. It is burned into the firmware of the Molex
LED and it cannot be changed. As the value is unmuttable, it can reliably be a source
for Classification Rules inside of ISE and for applying  Authorization Policy.
It is important to see the specific URL Format on the MUDURL. It can be used as well
by the MUD classification service.
```

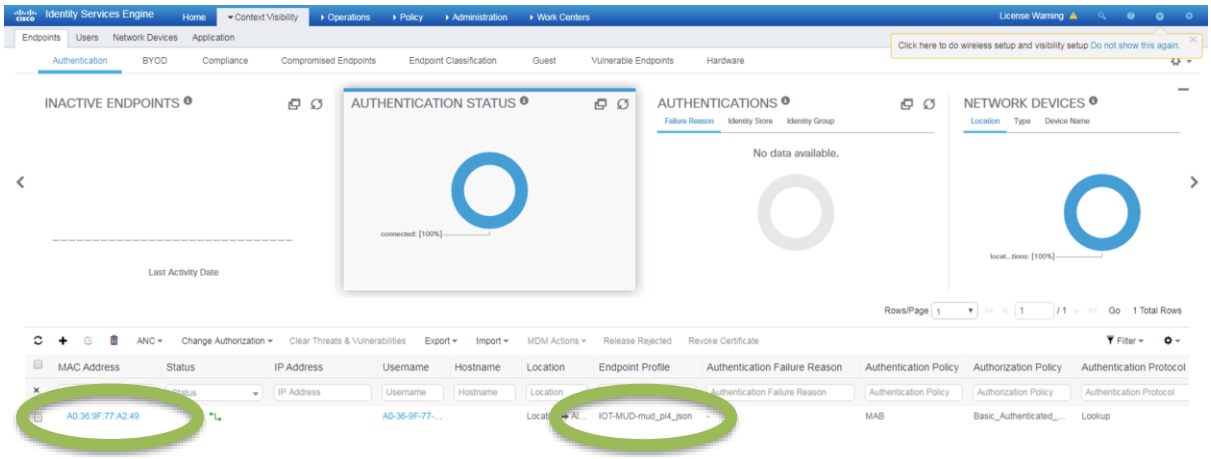
Step 3: Check if MAB passed authorization

```
C9300-SMU# show access-session interface g1/0/24 details
            Interface:  GigabitEthernet1/0/24
               IIF-ID:  0x1F6C03FC
          MAC Address:  a036.9f77.a249
         IPv6 Address:  fe80::a236:9fff:fe77:a249
         IPv4 Address:  Unknown
            User-Name:  A0-36-9F-77-A2-49
               Status:  Authorized
               Domain:  DATA
       Oper host mode:  multi-auth
     Oper control dir:  both
      Session timeout:  N/A
    Common Session ID:  6F03010A0000001B4089CDE5
      Acct Session ID:  0x00000003
               Handle:  0x5b000011
       Current Policy:  mud-mab-test
Server Policies:
Method status list:
       Method           State
          mab           Authc Success
C9300-SMU#
```

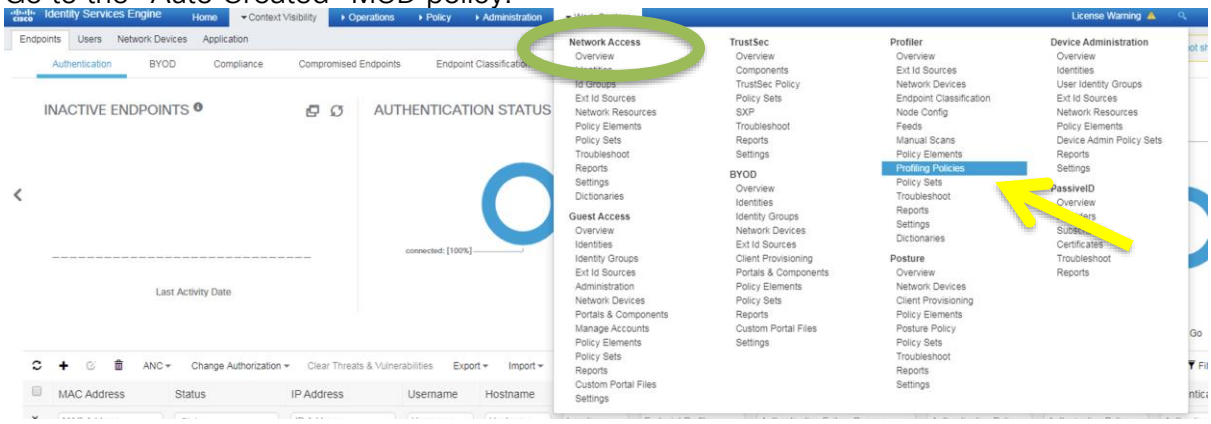
Task 3: Verify parameters send to ISE via RADIUS accounting

The endpoint should now be shown, as it was discovered by the Catalyst 9300 running device discovery.

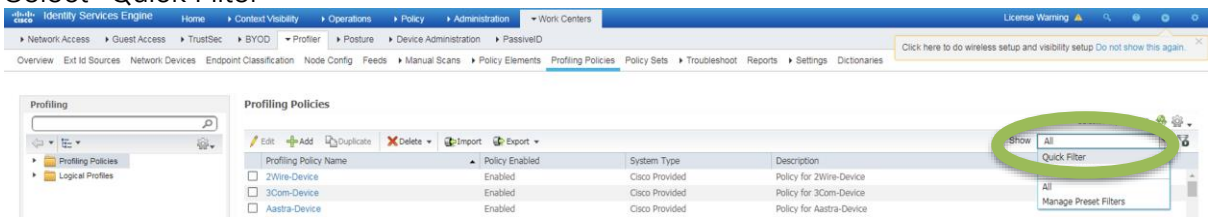




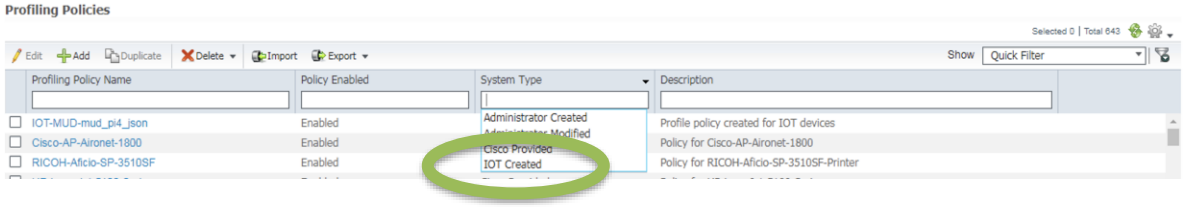
Go to the “Auto Created” MUD policy.



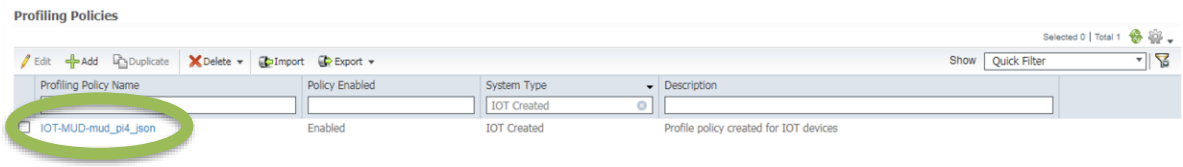
Select “Quick Filter”



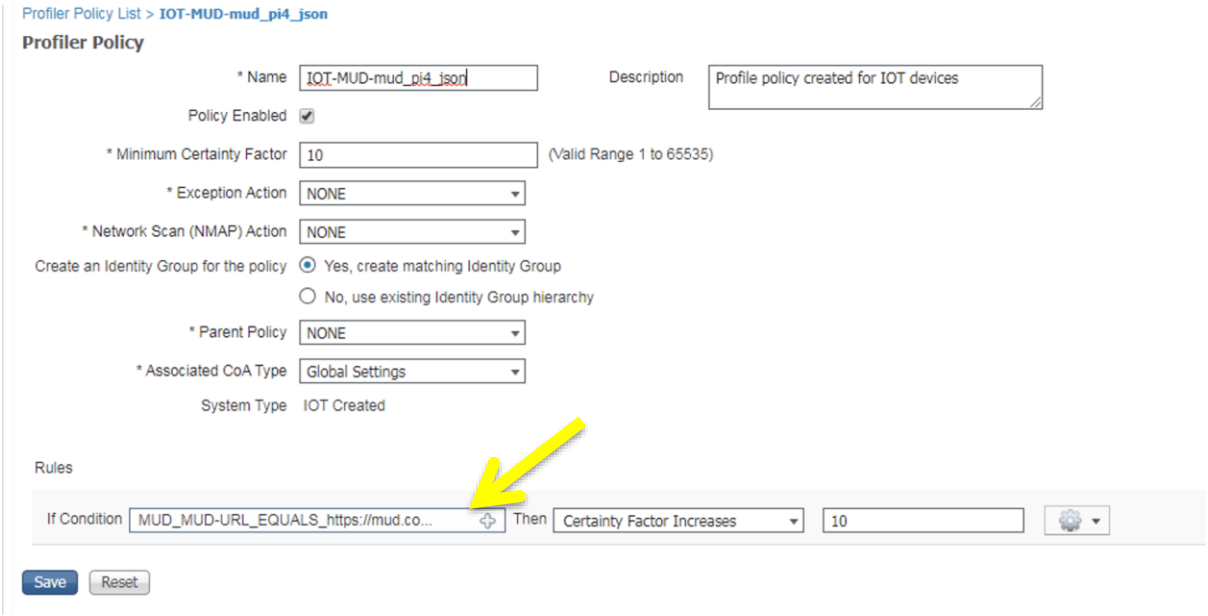
Under “System Type” select “IOT Created”



Click on “IOT-MUD …”



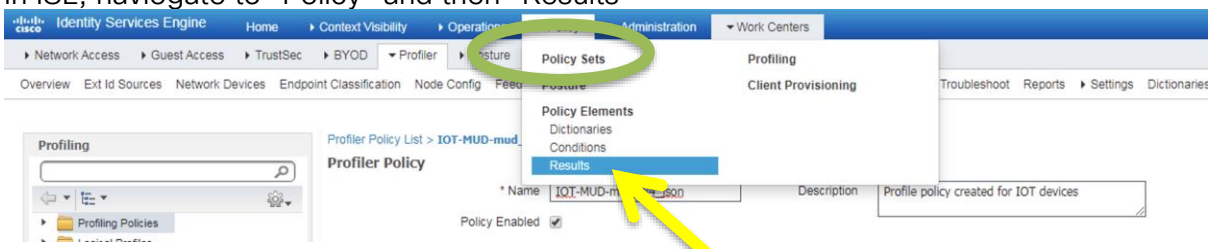
Check if the URL is the same as the custom TLV configured.



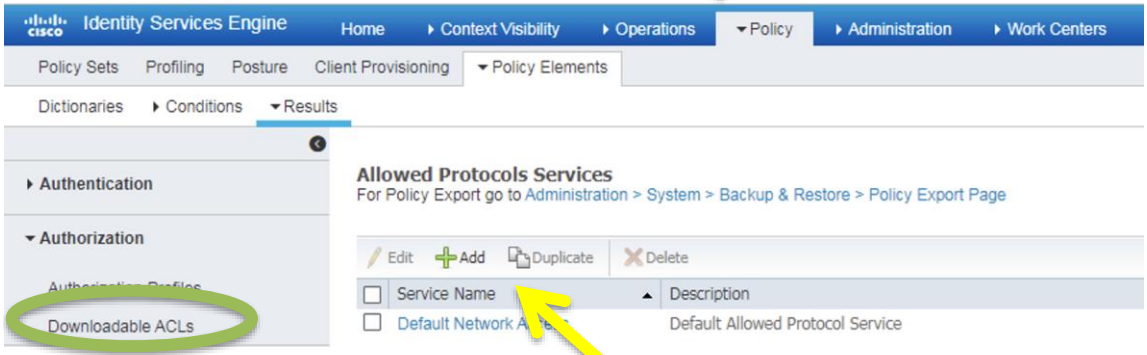
Task 4: Set Custom Authorization if MUD device is detected

Step 1: Create new Authorization Result for the MUD devices

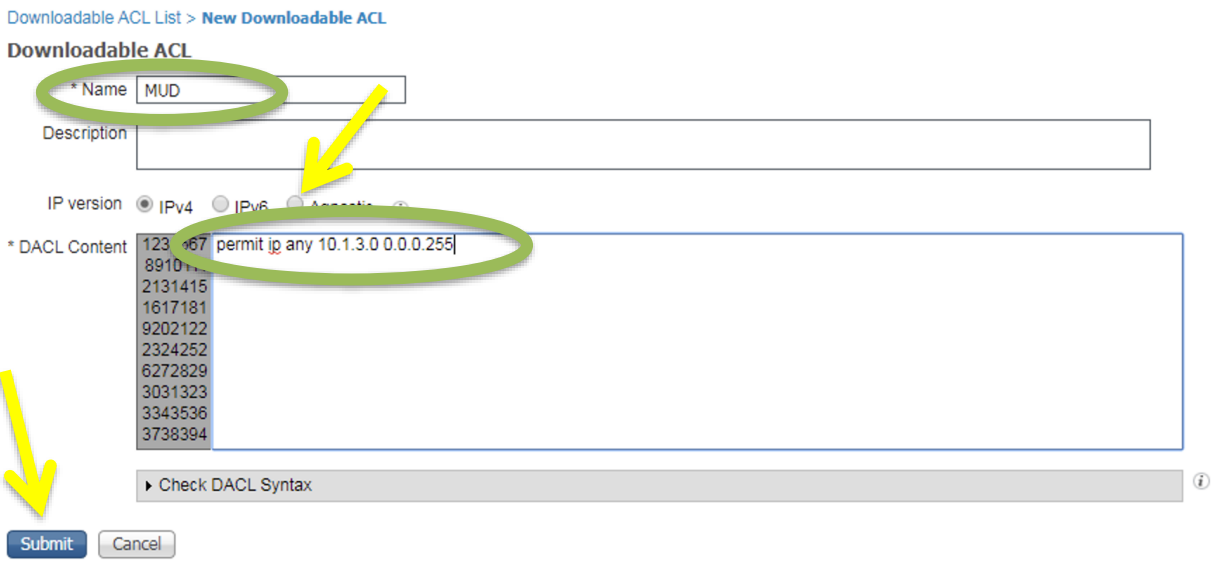
In ISE, naviogate to “Policy” and then “Results”



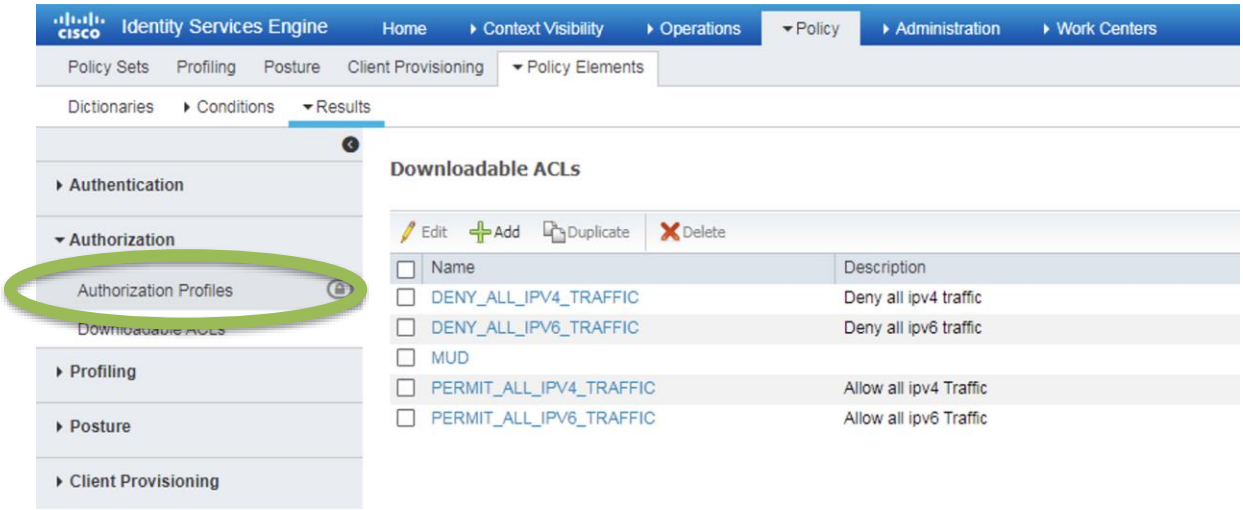
Select “Downloadable” ACL and click on “+ Add”



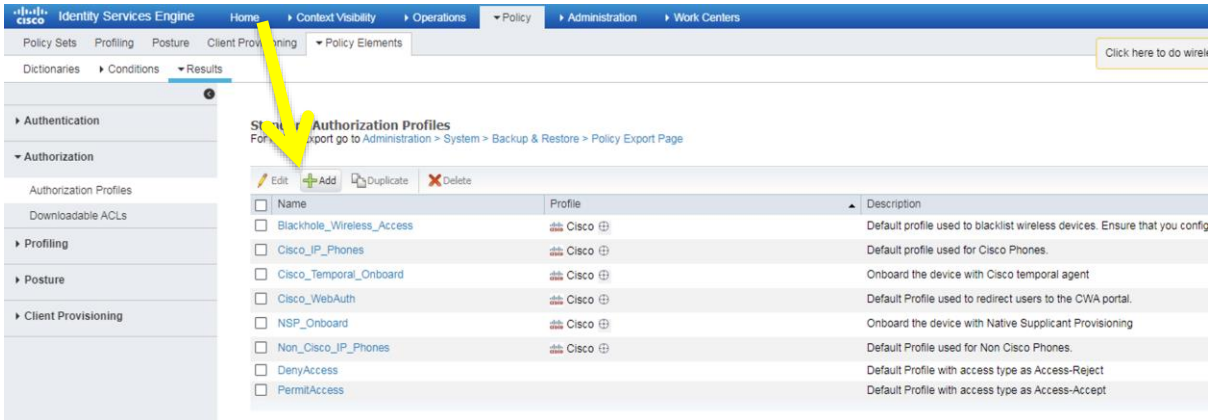
Enter name “MUD”. Enter the the following ACL rule and check it. “Source Net” should be “any” as it will be auto replaced by the switch later.
Select *“Agnostic”*
Type ACL: *permit ip any 10.1.3.0 0.0.0.255*
Press *“Submit”*



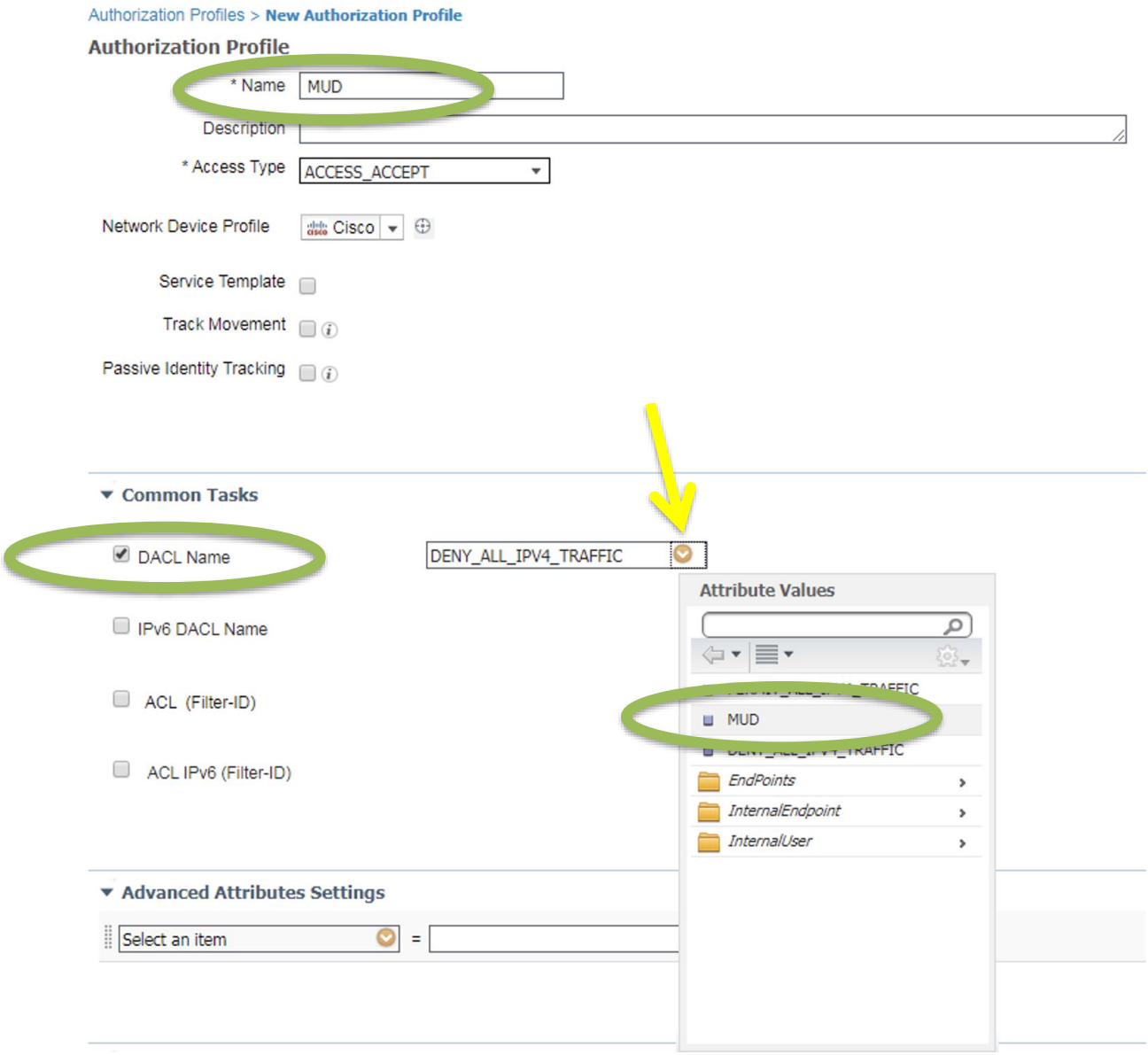
Create an Authorization result



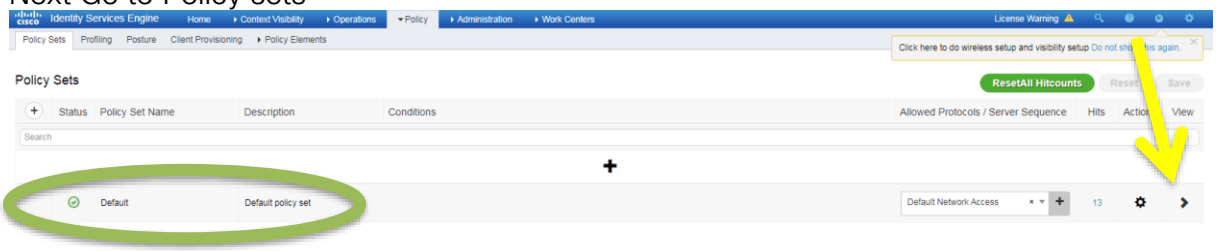
Click on "+ Add"



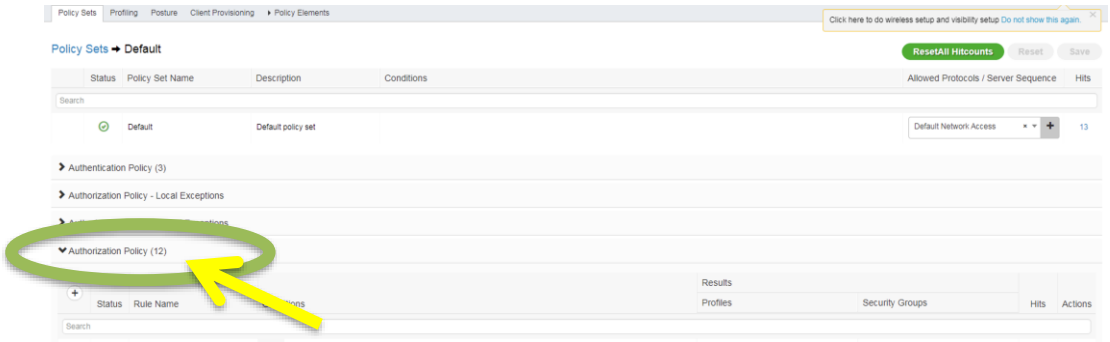
Enter the same Attributes:



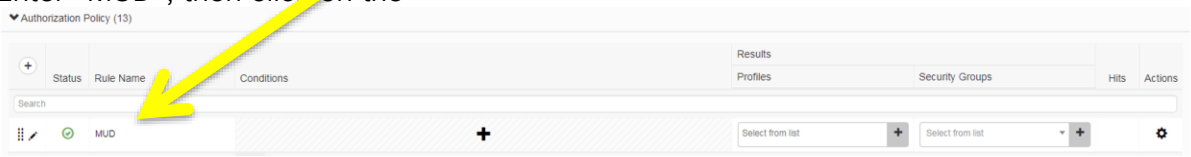
Click “Submit” at the bottom on the page.
Next Go to Policy sets



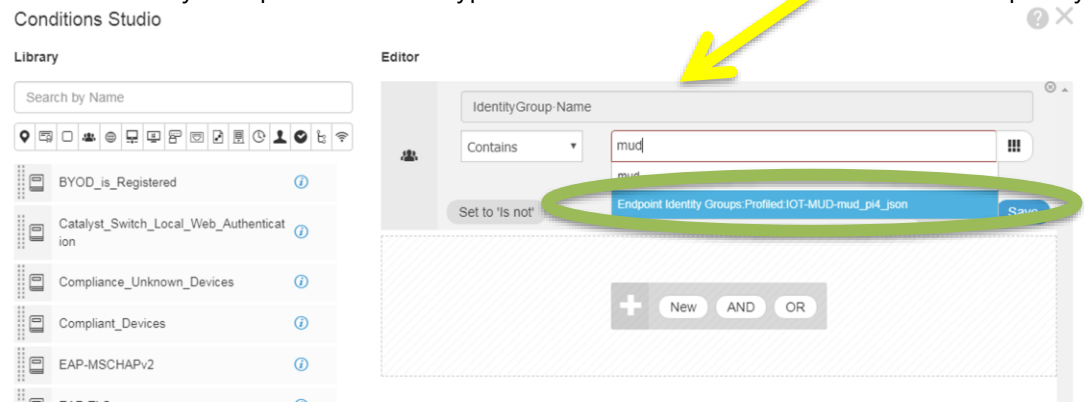
Click on “Authorization Policy” and the “+”



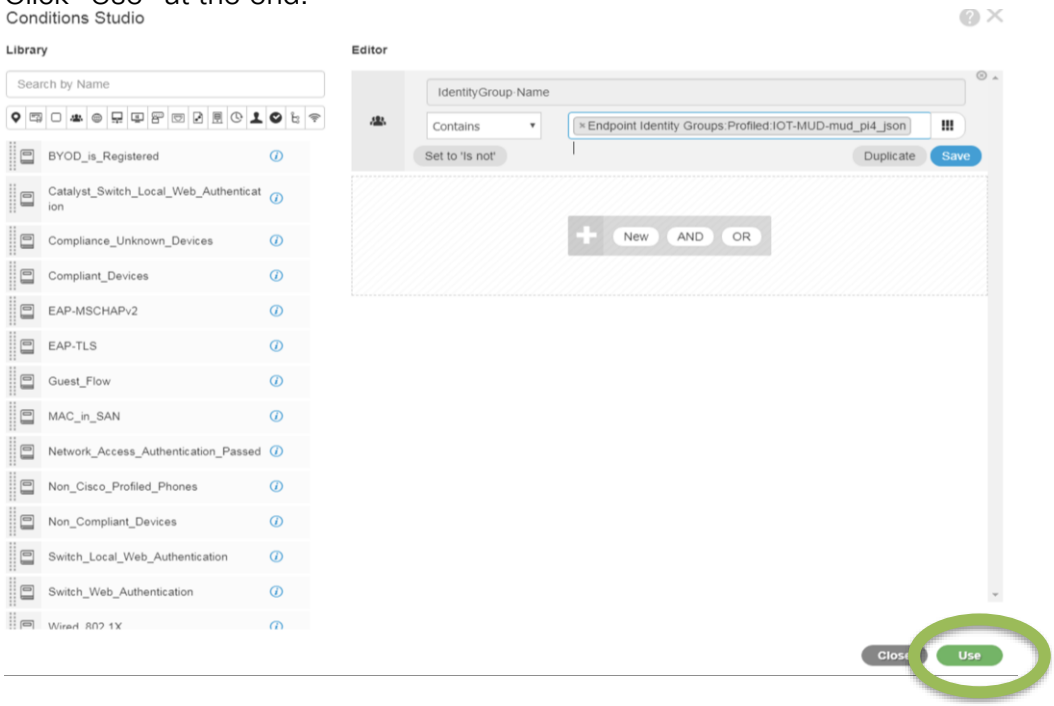
Enter “MUD”, then click on the “+”



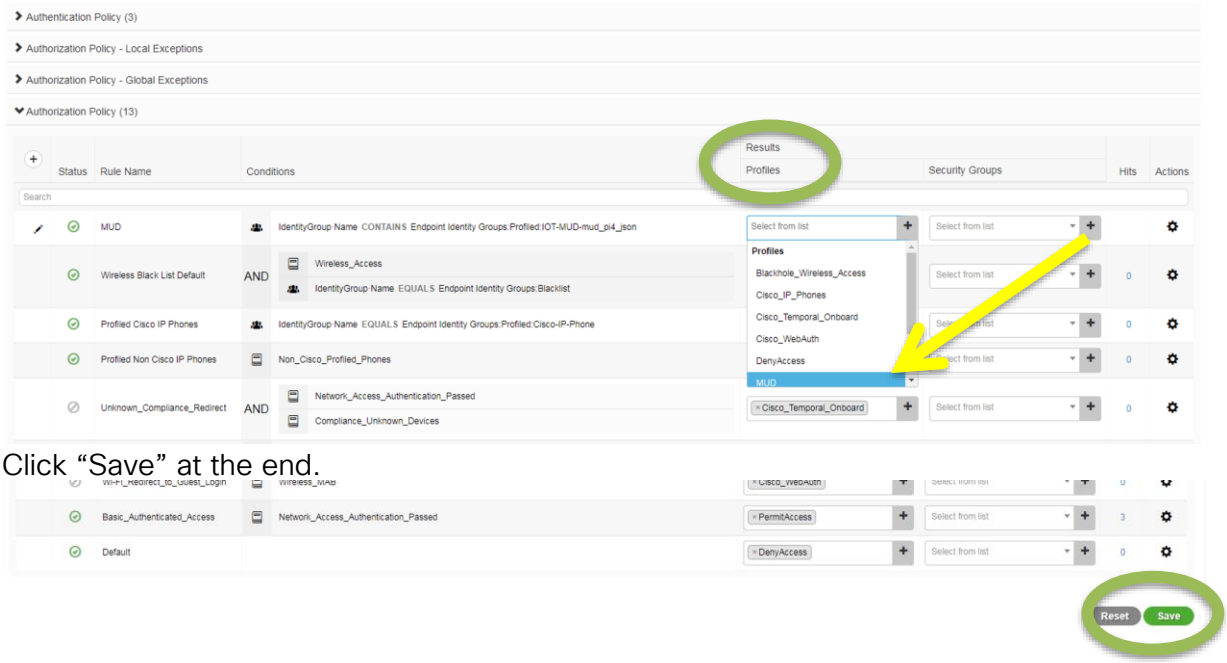
select “IdentityGroup Name” and type “mud”. Then select the auto created policy.



Click “Use” at the end.



Select “Result” as MUD



Click “Save” at the end.

Step 2: At the End, Trigger a new authorization to check the custom result

```
C9300-SMU# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9300-SMU(config)#int g1/0/24
C9300-SMU(config-if)#sh
C9300-SMU(config-if)#no sh
C9300-SMU(config-if)#end
C9300-SMU#show access-session interface g1/0/24 details
            Interface:  GigabitEthernet1/0/24
                IIF-ID:  0x16E09851
           MAC Address:  a036.9f77.a249
          IPv6 Address:  Unknown
          IPv4 Address:  Unknown
             User-Name:  A0-36-9F-77-A2-49
                Status:  Authorized
                Domain:  DATA
       Oper host mode:  multi-auth
     Oper control dir:  both
      Session timeout:  N/A
    Common Session ID:  6F03010A0000002041E78DC0
      Acct Session ID:  0x00000004
                Handle:  0xaf000016
       Current Policy:  mud-mab-test
Server Policies: → This is the result of the classification
              ACS ACL: xACSACLx-IP-MUD-57f6b0d3
Method status list:
       Method           State
          mab           Authc Success
C9300-SMU#
```

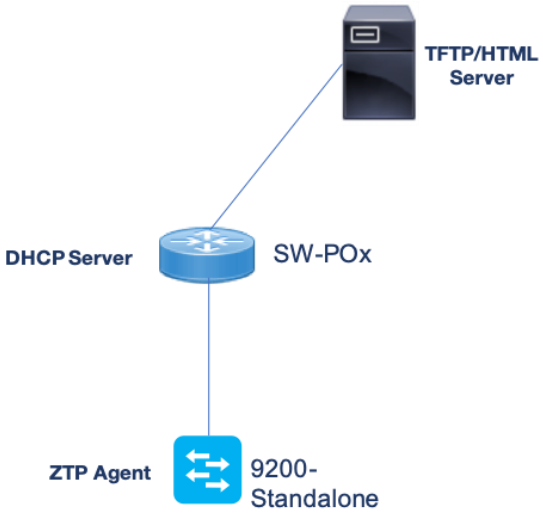
Scenario Zero Touch Provisioning (ZTP)

The Zero Touch Provisioning (ZTP) process is similar to Plug-n-Play (PnP), but it is designed to work with standard servers and uses protocols like TFTP and HTTP. PnP is the premium solution made possible with Cisco DNA Center, while ZTP is for the do-it-yourself customers who don't mind investing more time in configuring and maintaining the infrastructure required to bootstrap devices.

When a device that supports ZTP boots up and does not find the startup configuration (during a fresh install on Day Zero), the device enters the ZTP mode. The device locates a DHCP server, bootstraps itself with its interface IP address, gateway, and DNS server IP address, and enables Guest Shell. The device then obtains the IP address or URL of a TFTP server and downloads a Python script to configure the device.

Guest Shell provides the environment for the Python script to run. Guest Shell executes the downloaded Python script and configures the device for Day Zero. After Day Zero provisioning is complete, the Guest Shell instance is deactivated.

Network Diagram ZTP

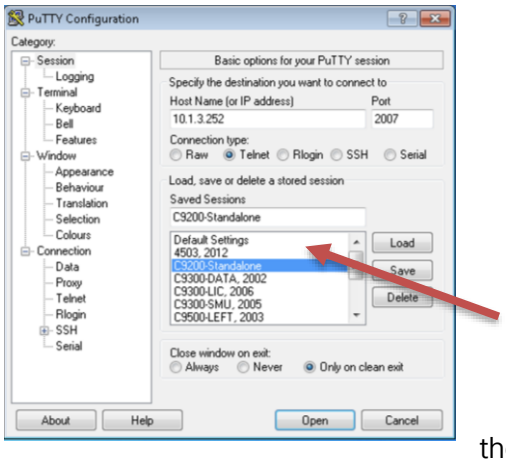


In this fairly simple topology, we have a Catalyst 9200 capable of running the ZTP Agent along with a switch that acts as a DHCP server and is a running TFTP server that stores the Python script that will be executed on the switch.

Task 1: Initialize the config for ZTP

Step 1: Device Connection

Connect to the CC9200-Standalone switch using Putty and verify the software version and connectivity to the DHCP and TFTP servers.



Once connected, verify that the management interface has a DHCP address via SW-P0<Pod#>

```
C9200-Standalone# show ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES unset  up                    up
GigabitEthernet0/0     10.1.3.120      YES DHCP   up                    up
GigabitEthernet1/0/1   unassigned      YES unset  down                  down
```

```
C9200-Standalone# show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
SW-POD[X]          Gig 0/0           137            R S I  WS-C4948- Gig 1/3
Total cdp entries displayed : 1
```

Now verify connectivity to TFTP server(10.1.3.106) that stores python script to be executed within the Guestshell.

```
C9200-Standalone# ping vrf Mgmt-vrf 10.1.3.106
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.3.106, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

Step 2: Verify DHCP Server configuration.

In Zero-Touch Provisioning, a DHCP server must be running on the same network as the new device that is being provisioned. ZTP is supported on both management ports and in-band ports.

When the new device is switched on, it retrieves the IP address information of the HTTP/TFTP server, where the Python script resides, and the folder path of the Python script from the DHCP server.

The DHCP server responds to DHCP discovery events with the following options:

Option 150 – (Optional) Contains a list of IP addresses that points to the HTTP/TFTP server on the management network that hosts the Python scripts to be run.

Option 67 – Contains the Python script file path on the HTTP/TFTP server. This option, also called the "bootfile name," tells the device which file to load and from where it is available.

Following are a few examples of how we can configure this on either the ISC DHCP Server or on the Cisco IOS DHCP Server.

Cisco DHCP Server Configuration

```
ip dhcp pool Mgmt-Vlan3
  network 10.1.3.0 255.255.255.0
  default-router 10.1.3.1
  domain-name cisco.com
  dns-server 10.1.3.1
  option 150 ip 10.1.3.106
  option 67 ascii simpleztp.py
```

The configuration example for the Linux ISC DHCP dhcpd.conf is below:

```
subnet 10.1.3.0 netmask 255.255.255.0 {
option bootfile-name "http://10.3.1.106/simpleztp.py"; }
```

Step 3: Initiate ZTP process.

Now that the prerequisites for ZTP are met, the device is needs to be reloaded once any previous configuration is removed. This is to ensure that the Day0 ZTP process is initialized once the switch boots. This emulates a new, unconfigured device that is ready to be brought onto the network via ZTP.

Erase the configuration and reload the device as follows:

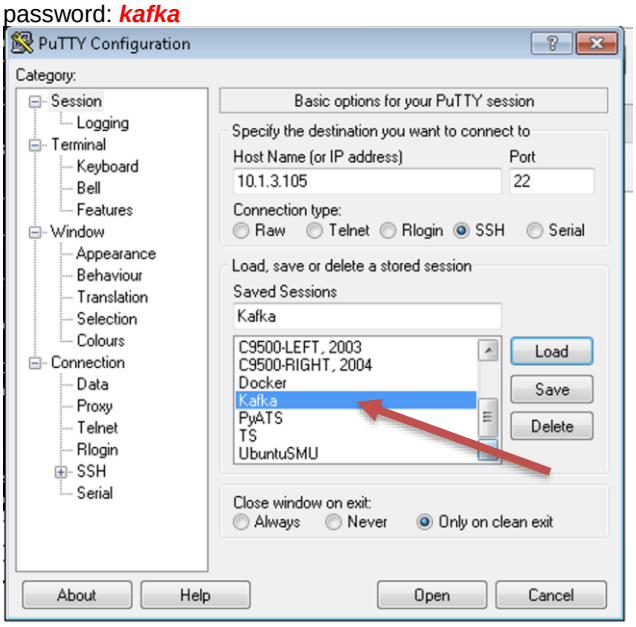
```
C9200-Standalone# write erase
Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]
[OK]
Erase of nvram: complete
C9200-Standalone#
*Sep 26 21:00:13.522: %SYS-7-NV_BLOCK_INIT: Initialized the geometry of nvram

C9200-Standalone# reload
System configuration has been modified. Save? [yes/no]: no
*Sep 27 00:01:06.275: %SYS-5-CONFIG_P: Configured programmatically by process Exec from
```

```
console as console
Reload command is being issued on Active unit, this will reload the whole stack
Proceed with reload? [confirm]
*Sep 27 00:01:13.220: %SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload
Command.
Chassis 1 reloading, reason - Reload command
```

Step 4: Review the script to be executed by the ZTP process.

Review the simpleztp.py file specified on DHCP option 67 or bootfile.

Login to “Kafka” Ubuntu from Putty via SSH using
username: ***kafka***
password: ***kafka***



Connect to the TFTP server 10.1.3.106 file specified on DHCP option 150 and do a get for *simpleztp.py*

```
kafka@ott-kafka-1:~$ tftp 10.1.3.106
tftp> get simpleztp.py

Received 465 bytes in 0.0 seconds
tftp> quit
```

Once the file has been successfully been received via TFTP, open the file to examine its content

```
kafka@ott-kafka-1:~$ cat simpleztp.py
       print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"
       # Importing cli module
       import cli
       print "Configure vlan interface, gateway, aaa, and enable netconf-yang\n\n"
       cli.configurep(["hostname 9200_ZTP_Configured", "end"])
       cli.configurep(["netconf-yang", "end"])
       print "\n\n *** Executing show ip interface brief  *** \n\n"
```

```
cli_command = "sh ip int brief"
cli.executep(cli_command)
print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

This file uses the Python API to set the hostname and configure access to the device over the NETCONF and RESTCONF programmatic interfaces, and configures some other device features.

To interact with CLI on the switch, three Python modules are available that are the API between Guest Shell and Catalyst 9000 Series Switches:

- *cli.cli*: This function takes an IOS command as an argument, runs the command through the IOS parser, and returns the resulting text.
- *cli.execute*: This function executes a single EXEC command and returns the output; however, does not print the resulting text. No semicolons or newlines are allowed as part of this command. Use a Python list with a for-loop to execute this function more than once
- *cli.configure*: This function configures the device with the configuration available in commands. It returns a list of named tuples that contains the command and its result.

Task 2: Verify ZTP with Python script results

Step 1: Verify ZTP on Catalyst 9000 Series Switch

DO NOT TOUCH the console up until "ZTP Day0 Python Script Execution Complete" message is seen as it might interrupt ZTP/Boot process.

Login back to CC9200-Standalone console and observe console logs.

```
Would you like to enter the initial configuration dialog? [yes/no]:
day0guestshell installed successfully
Current state is: DEPLOYED
day0guestshell activated successfully
Current state is: ACTIVATED
day0guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully
HTTP server statistics:
Accepted connections total: 0
 *** Sample ZTP Day0 Python Script ***
Configure vlan interface, gateway, aaa, and enable netconf-yang
Line 1 SUCCESS: hostname 9200_ZTP_Configured
Line 2 SUCCESS: end
Line 1 SUCCESS: netconf-yang
Line 2 SUCCESS: end
 *** Executing show ip interface brief  ***
```

```
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES unset  up                    up
GigabitEthernet0/0     10.1.3.128      YES DHCP   up                    up
GigabitEthernet1/0/1   unassigned      YES unset  down                  down
GigabitEthernet1/0/2   unassigned      YES unset  down                  down
GigabitEthernet1/0/3   unassigned      YES unset  down                  down
GigabitEthernet1/0/4   unassigned      YES unset  down                  down
GigabitEthernet1/0/48  unassigned      YES unset  down                  down
GigabitEthernet1/1/1   unassigned      YES unset  down                  down
GigabitEthernet1/1/2   unassigned      YES unset  down                  down
GigabitEthernet1/1/3   unassigned      YES unset  down                  down
GigabitEthernet1/1/4   unassigned      YES unset  down                  down
Te1/1/1                unassigned      YES unset  down                  down
Te1/1/2                unassigned      YES unset  down                  down
Te1/1/3                unassigned      YES unset  down                  down
Te1/1/4                unassigned      YES unset  down                  down
 *** ZTP Day0 Python Script Execution Complete ***
Guestshell destroyed successfully
Press RETURN to get started!
```

Observe the highlighted part of the console logs, wherein after receiving option 67 the switch initiates the ZTP agent and runs Guest Shell to execute the Python script from

Step 4. Once script execution is complete, the Guest Shell is c and day0 onboarding is complete.

Now, login to the switch to verify that configurations have taken effect by verifying that NETCONF/YANG process are up and running as configured in the script.

```
9200_ZTP_Configured# show netconf-yang status
netconf-yang: enabled
netconf-yang ssh port: 830
netconf-yang candidate-datastore: disabled
```

The Python script that was executed is stored locally as downloaded_script.py and it continues to be stored on the flash.

```
9200_ZTP_Configured# more downloaded_script.py
print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"
# Importing cli module
import cli

print "Configure vlan interface, gateway, aaa, and enable netconf-
yang\n\n"
cli.configurep(["hostname 9200_ZTP_Configured", "end"])
cli.configurep(["netconf-yang", "end"])

print "\n\n *** Executing show ip interface brief  *** \n\n"
cli_command = "sh ip int brief"
cli.executep(cli_command)

print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

Summary

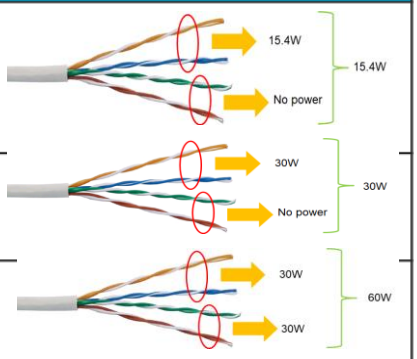
Guest Shell and ZTP offer a flexible, programmable Day0 device onboarding capability by utilizing the Catalyst 9000's Python API. This API allows the Guest Shell to send commands to the IOS XE operating system. What kind of commands? Show commands are supported with the Python CLI module cli.cli. Show commands are great for displaying information about the device but are limited when it comes to making device configuration changes. To really harness the power of the Python API, the cli.execute and cli.configure modules provide a great deal of flexibility when it comes to device configuration. We can interact with the device through Python using the traditional "configure terminal" ("conf t") interface and even send exec commands as needed. All the power of the CLI, now with the flexibility of Python.

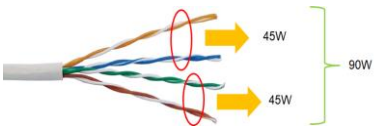
Scenario Configure 90W based on 802.3bt

The IEEE 802.3bt standard, which was ratified in September 2018, enables *delivery of up to 90W* to a powered device using four pairs of Category 5e and above cables. It Introduces *4 new Classes, class 5 to class 8*, for PDs (Powered Devices), where PSE (Power Sourcing Equipment) output is between 45W – 90W and PD input range is between 40W – 73W. It also Introduces 2 new Types for PSEs and PDs – Type 3(60W) and Type 4 (90W). It enables support for *Dual Signature PDs and Single Signature PDs*, and it supports *Power Demotion* to handle scenarios where a Type 4 PD is connected to a Type 3 PSE.

IEEE 802.3bt Standard

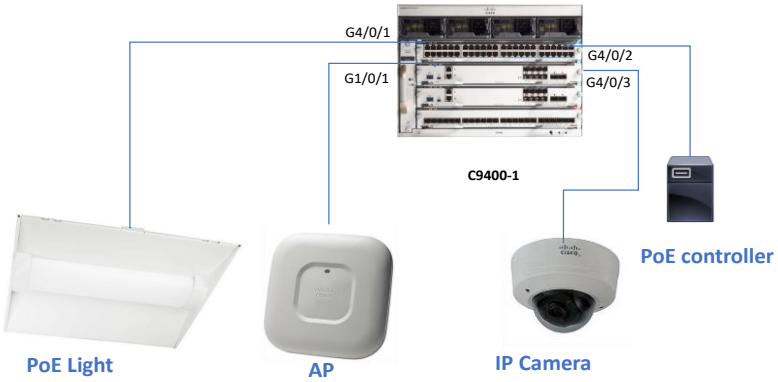
Power Over Ethernet Categorization

PoE Standard	Type	Class	Max Power	Power over wire pairs
Cisco Phone Discovery	-	-	15.4W	
802.3af (PoE)	1	0-3	15.4W	
802.3at (PoE+)	2	4	30W	
Cisco UPOE	(2)	(4)	60W	
802.3bt (60W, 90W)	3	5, 6	60W	
	4	7, 8	90W	



Cisco Introduced its UPOE+ (90W) capable line card (*C9400-LC-48H*) for Cisco Catalyst 9400 series switches, which is compatible with all C9400 chassis types and supervisors. It supports IEEE 802.3bt standards and all previous IEEE power over ethernet standards, like 802.3at, 802.3af and Cisco pre-standard devices. The Cisco Catalyst UPOE line cards (*C9400-LC-48UX and C9400-LC-48U*) can be converted to 803.3bt mode via CLI to enable support for Class 5 and 6 (up to 60W) endpoints / PDs. Note: UPOE line cards are *Type 3 PSE and support up to 60W per port or Class 6*

Network Diagram IEEE 802.3bt Standard

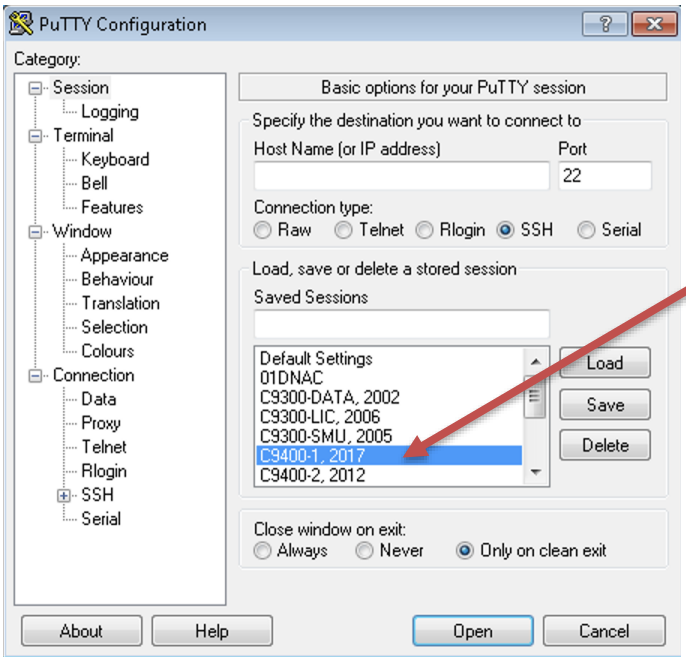


In this lab, we are using a Cisco Catalyst 9404 as a PSE (Power Sourcing Equipment) and an IEEE 802.3bt compliant PoE light and older PoE devices as the Powered Devices (PDs). We will use a 3rd-party PoE controller to manage the PoE power and observe the power consumption on the Cisco Catalyst 9400 switch.

Task 1: Initialize Catalhyst 9400 switch for 802.3bt

Step 1: Device Connection

Connect to the C9400-1 switch using Putty and verify the configuration using show commands via CLI



Once connected, run show commands to verify the modules and power budget. Please see the highlighted parts in the below output. “Show module” will show us what type of line cards we have and other information like MAC addresses, serial numbers and model number. C9400-LC-48UX is a 48 port Cisco UPOE (60W) and Multigigabit (mGig) line card and C9400-LC-48H is a 48 port UPOE+ (90W) line card based on the IEEE 802.1bt standard. The setttings of C9400-LC-48UX can be changed to meet the IEEE 802.1bt standard for class 5 and class 6 devices.

“*show power*” will show us how many power supplies there are in the system , how much system power is being consumed, and how much PoE power is available on the system. With the information of how much inline power is available we can add PoE devices accordingly.

“*show power*” will show us how many power supplies are there and how much power is the system power consumption and how much PoE power is available on the system, with a lot more information. With the information of how much inline power is available we can add PoE devices accordingly.

```
C9400-1# show module
Chassis Type: C9404R
Mod Ports Card Type                                     Model           Serial No.
---+-----+-----------------------------------------+--------------+--------------
1    48    48-Port UPOE w/ 24p mGig 24p RJ-45          C9400-LC-48UX     JAE224600LK
2    12    Supervisor 1 XL Module with 25G             C9400-SUP-1XL-Y   JAE22350JVB
4    48    48-Port 90W BT 10/100/1000 (RJ-45)          C9400-LC-48H      JAE231008JS
Mod MAC addresses                       Hw    Fw           Sw                 Status
---+-------------------------------+----+------------+------------------+--------
1    F80F.6FBC.0404 to F80F.6FBC.0433 1.0   16.12.1r      16.12.01           ok
2    F80F.6FCB.596C to F80F.6FCB.5977 1.0   16.12.1r      16.12.01           ok
4    706D.15B3.1A24 to 706D.15B3.1A53 0.5   16.12.1r      16.12.01           ok
```

```
Mod Redundancy Role     Operating Redundancy Mode Configured Redundancy Mode
---+--------------------+-------------------------+---------------------------
2   Active               non-redundant              sso
C9400-1#

C9400-1# show power
Power                                                                Fan States
Supply  Model No               Type  Capacity  Status        1     2
------  --------------------  ----  --------  ------------  ----------
PS1     C9400-PWR-3200AC       ac    3200 W    active        good  good
PS2     C9400-PWR-3200AC       ac    3200 W    active        good  good
PS Current Configuration Mode : Combined
PS Current Operating State    : Combined
Power supplies currently active    : 2
Power supplies currently available : 2
Power Summary          Maximum
 (in Watts)    Used    Available
-------------  ------  ---------
System Power   1475    1475
Inline Power   152     4925
-------------  ------  ---------
Total          1627    6400
```

Step 2: Convert the UPOE line card in slot 1 into BT mode.

We will use the following commands to convert the UPOE line card in slot 1 to IEEE 802.bt mode. The module continues to provide 60W but now is IEEE 802.1bt compliant.

```
C9400-1# conf t
C9400-1(config)# hw-module slot 1 upoe-plus
C9400-1(config)# end
C9400-1#
```

The system will reload the only slot where IEEE 802.3bt mode is configured, which is slot1 in our case. This will change the default IEEE 802.3at mode to IEEE 802.3bt mode, and POE devices are expected to go down and come back up again once the line card reloads.

Note: IEEE 802.3bt compliance for UPOE line cards was introduced in IOS XE 16.11.1 for Cisco Catalyst 9400 and IOS XE 16.12.1 for Cisco Catalyst 9300 UPOE capable switches. The UPOE+ (90W) line cards is IEEE 802.3bt compliant by default and requires IOS XE 16.12.1 or later releases.

Step 3: We will check how many PDs are connected to the switch.

Optional CLIs:

```
show power inline
show power inline upoe-plus
```

Note: These two commands will give the output of all the ports that are PoE capable and will provide almost the same information. Feel free to observe the output of these two CLIs. "*show power inline upoe-plus*" will give more information that includes the device

names in addition to other information. We can see the output in below screenshot of "*show power inline upoe-plus*".

```
C9400-1#show power inline upoe-plus

Available:4925.0(w)  Used:152.2(w)  Remaining:4772.8(w)

Codes: DS - Dual Signature device, SS - Single Signature device
       SP - Single Pairset device

Interface   Admin  Type Oper-State       Power(Watts)      Class   Device Name
            State       Alt-A,B       Allocated Utilized  Alt-A,B
----------- ------ ---- ------------- --------- --------- ------- -----------

Gi1/0/1     auto   n/a  off           0.0       0.0       n/a
Gi1/0/2     auto   SS   on,off        16.8      6.1       4       AIR-CAP3702E-A-K9
Gi1/0/3     auto   n/a  off           0.0       0.0       n/a
Gi1/0/4     auto   n/a  off           0.0       0.0       n/a
Gi1/0/5     auto   n/a  off           0.0       0.0       n/a
Gi1/0/6     auto   n/a  off           0.0       0.0       n/a
Gi1/0/7     auto   n/a  off           0.0       0.0       n/a

C9400-1#
```

In this lab we will use "*show power inline upoe-plus | inc on*", which will show us the devices that are on and actually consuming power. It is important to note the highlighted parts on the above screenshot as we would see which *interface* the device is connected to, what kind of *state* is it in, the *type* of device, its *operation status* and most importantly *power allocated, power utilized* and the *device class*

```
C9400-1# show power inline upoe-plus | inc  on
Gi1/0/2     auto   SS   on,off         16.8      6.1       4       AIR-CAP3702E-A-K9
Gi4/0/2     auto   SS   on,on          90.0      0.8       8       Ieee PD
Gi4/0/3     auto   SS   on,off         15.4      4.6       3       Ieee PD
Gi4/0/4     auto   SS   on,off         30.0      14.3      4       Ieee PD
Totals:             4    on          152.2     25.8
```

We can see in the output two encircled values and see that the Admin State of all the PoE connected interfaces is "auto". Also, the power allocated, and power consumed by each device is different. In this lab we will do changes on G1/0/4, where an 802.3bt (90W) device is connected. We can see the device has negotiated as a class 8 device, the power consumed is 0.8W as of now, and the maximum power that the device can get is 90W. Anything above the allocated power by the PSE, would result in an imax error.

Step 4: Check UPoE Plus

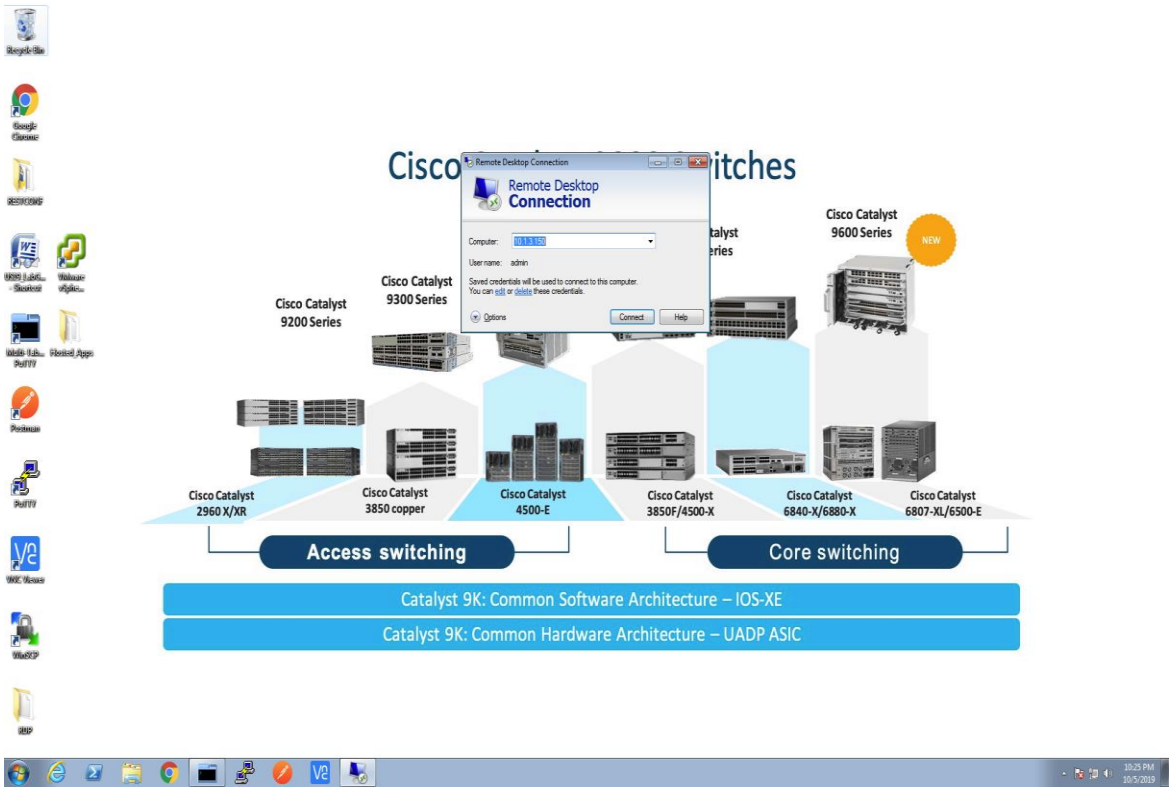
Now issue a command "*show power inline upoe-plus gigabitEthernet 4/0/2*", which will only show us the values of the specific interface G4/0/2.
Below screen show shows the counter to look, while the value will vary.

```
C9400-1#show power  inline upoe-plus gigabitEthernet 4/0/2
Codes: DS - Dual Signature device, SS - Single Signature device
       SP - Single Pairset device

Interface   Admin  Type Oper-State       Power(Watts)      Class   Device Name
            State       Alt-A,B       Allocated Utilized  Alt-A,B
----------- ------ ---- ------------- --------- --------- ------- -----------
```

```
Gi4/0/2      auto    SS    on,on          90.0       8.6        8        Ieee PD
C9400-1#
```

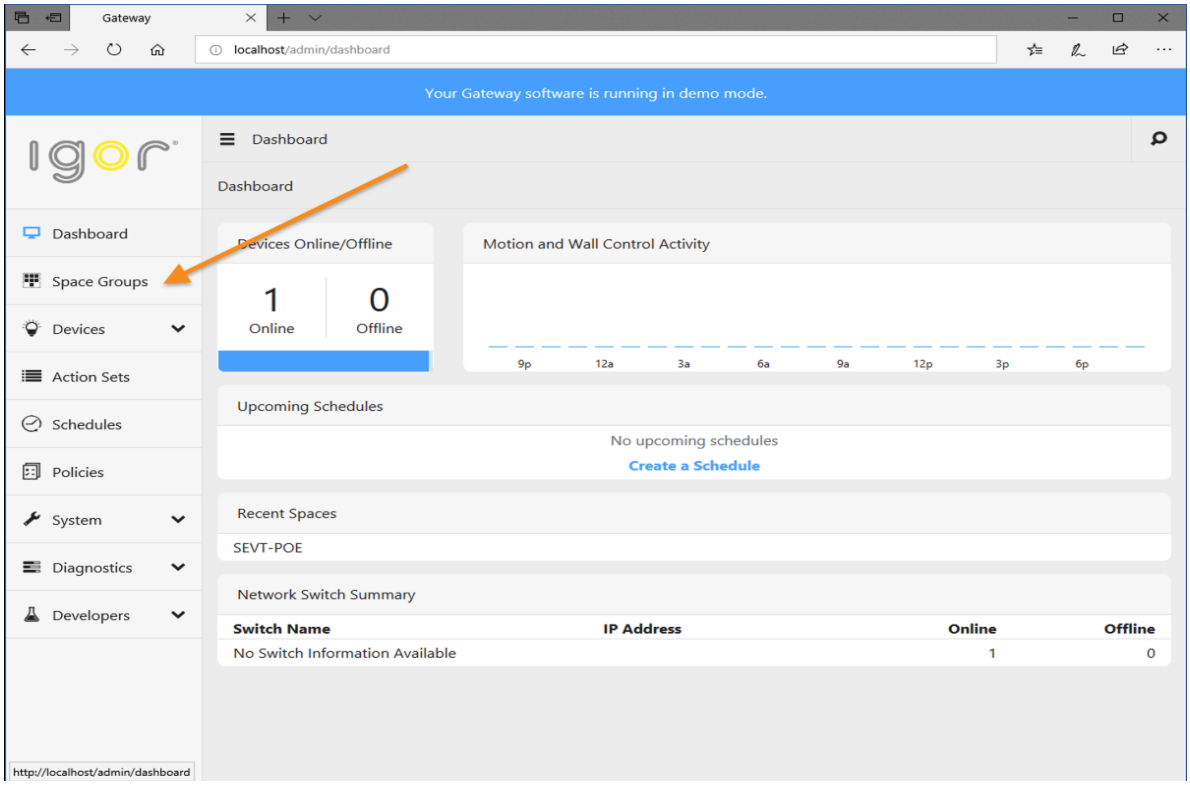
Task 2: Use endpoints to see the Power Conumption

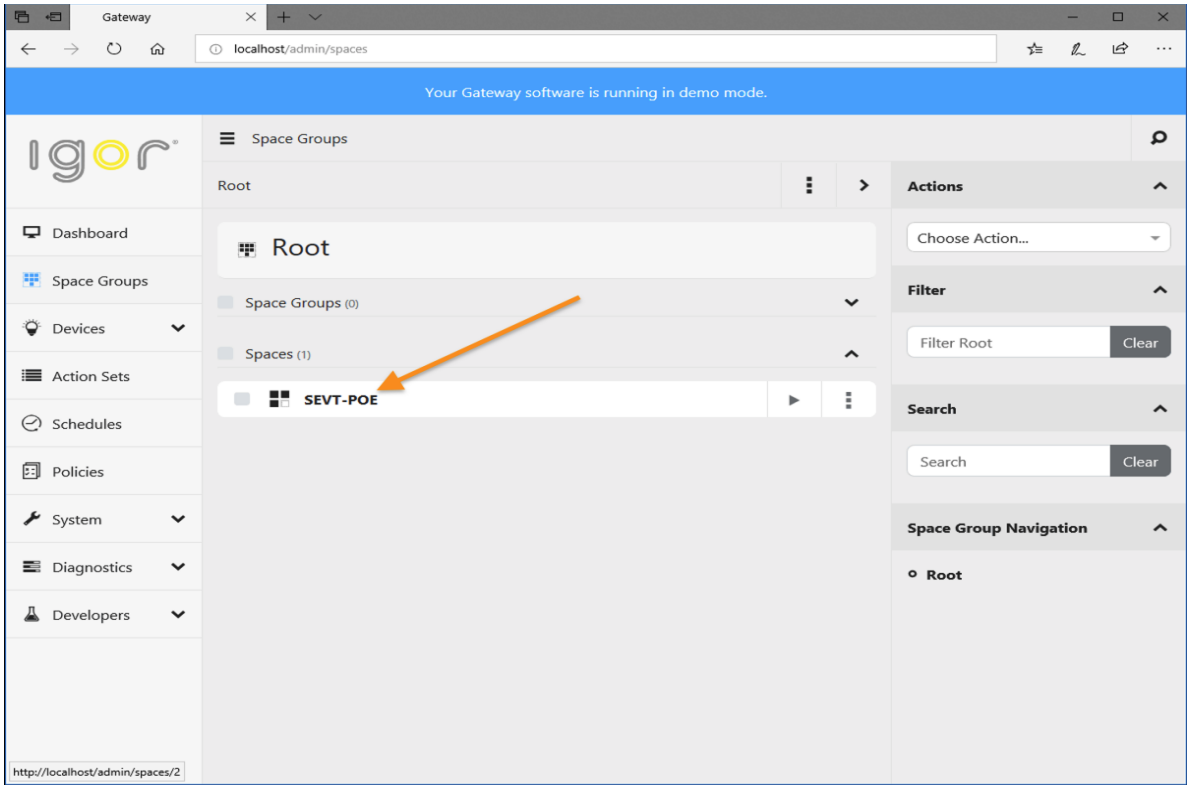
Open a remote desktop connection by clicking on the shortcut on the desktop and use the following *IP: 10.1.3.150*
Username: *admin*
Password: *cisco*



Once logged in, double click on IGOR Gateway software. When prompted for username and password, please use **admin** for username and **cisco** for password. This software is used to monitor and control the POE light. This could be any vendor POE light and software. The browser will open the software page, and we will increase and decrease the light intensity and would check the power consumption on the switch side.

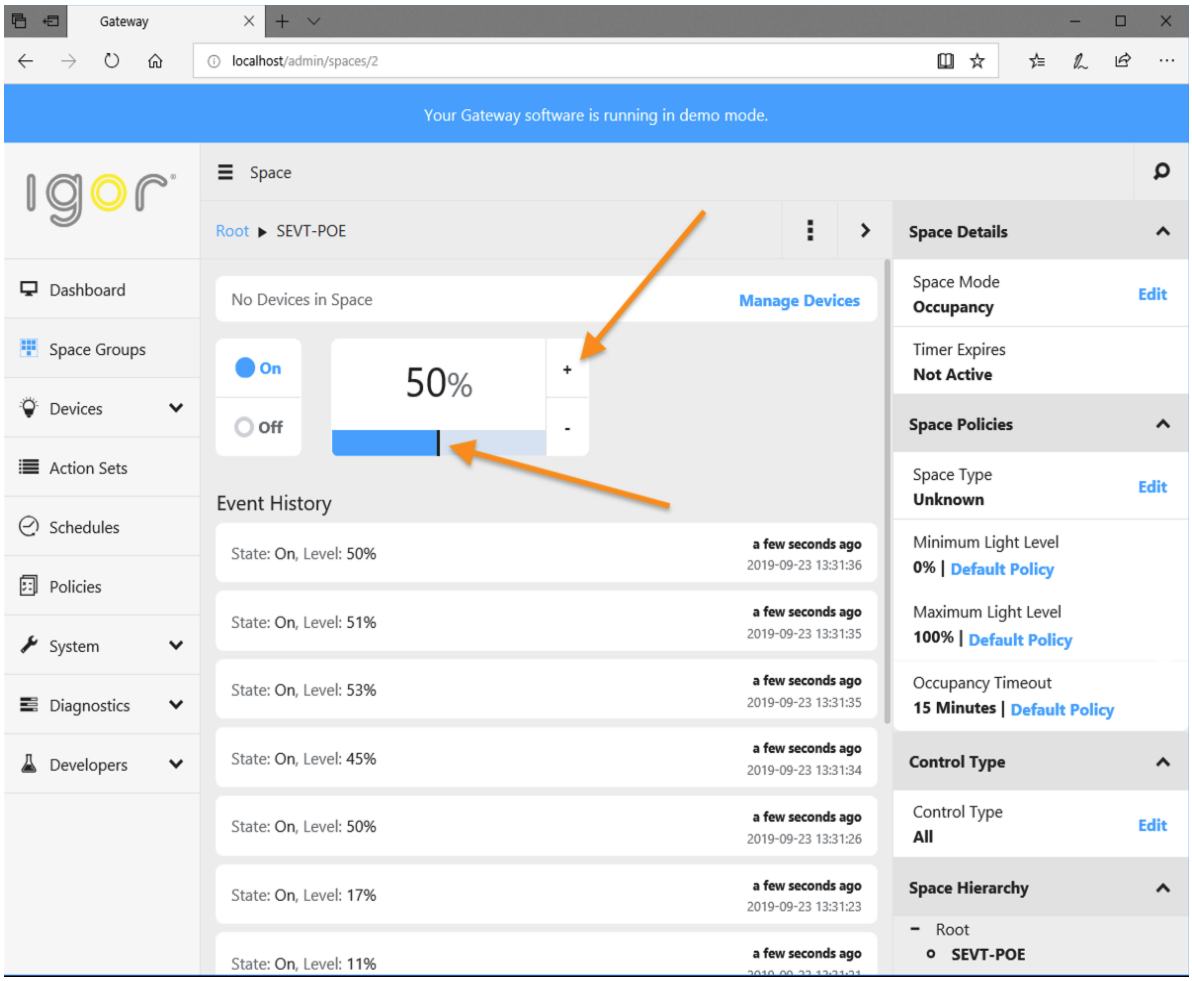
Step 1: Click on Spaces Groups tab on the left of the Gateway application and click on SEVT-POE.





Step 2: Change power consumption

Increase the power either by using + sign or the scroll it to 50% or close to it and see the power consumption on G4/0/2 by using "*show power inline upoe-plus gigabitEthernet 4/0/2*"

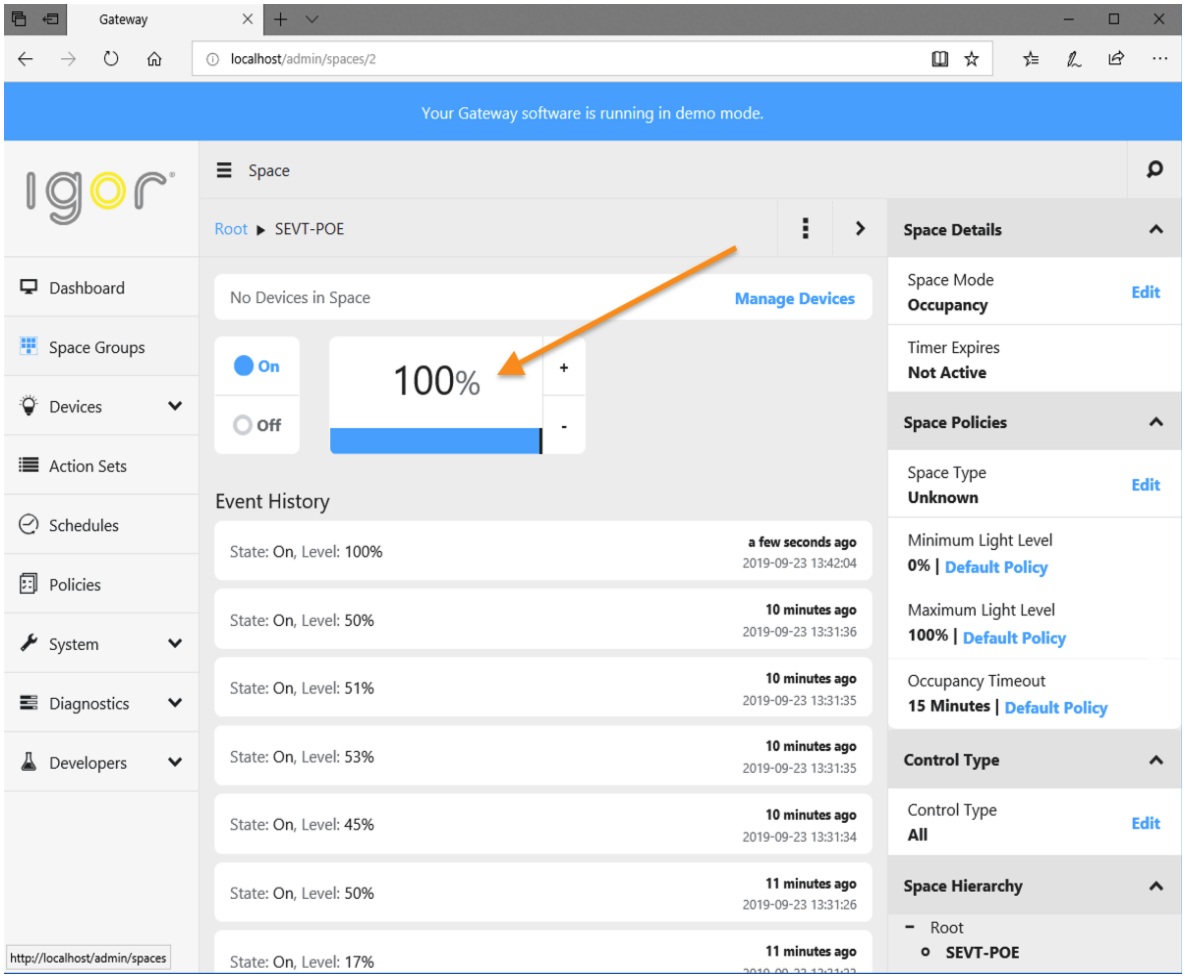


Now go back to the putty session that was already opened and issue the same command or use up arrow and press enter.

We would see the increase in the power utilized on G4/0/2

```
C9400-1#show power  inline upoe-plus gigabitEthernet 4/0/2
Codes: DS - Dual Signature device, SS - Single Signature device
       SP - Single Pairset device

Interface   Admin  Type Oper-State       Power(Watts)     Class   Device Name
            State       Alt-A,B       Allocated Utilized  Alt-A,B
----------- ------ ---- ------------- --------- --------- ------- -----------

Gi4/0/2     auto   SS   on,on         90.0      39.0      8       Ieee PD
C9400-1#
```

Keep on changing the power and increase it to maximum and see the output on the switch.




```
C9400-1#show power  inline upoe-plus gigabitEthernet 4/0/2
Codes: DS - Dual Signature device, SS - Single Signature device
       SP - Single Pairset device

Interface   Admin  Type Oper-State       Power(Watts)     Class   Device Name
            State       Alt-A,B       Allocated Utilized  Alt-A,B
----------- ------ ---- ------------- --------- --------- ------- -----------

Gi4/0/2     auto   SS   on,on         90.0      84.2      8       Ieee PD
C9400-1#
```

We can see that the switch is able to provide upto 85 watts of power over ethernet.
NOTE: Please bring the power back to 10% from the software.

Step 3: Allocating static power to the devices that are not BT compatible.

On the switch, apply the following CLI and allocate the static power to the device.

```
C9400-1(config)#
< Copy Paste in exec mode >
```

```
show run interface gigabitEthernet 4/0/2

C9404-1# show run interface gigabitEthernet 4/0/2
Building configuration...
Current configuration : 88 bytes
!
interface GigabitEthernet4/0/2
 switchport access vlan 3
 switchport mode access
end

C9400-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C9400-1(config)#int g4/0/2
C9400-1(config-if)#power inline static max ?
  <4000-90000>  milli-watts

C9400-1(config-if)#power inline static max 90000
```

Now do show run interface GigabitEthernet4/0/1 and see the output.

```
C9404-1# show run interface gigabitEthernet 4/0/2
Building configuration...
Current configuration : 109 bytes
!
interface GigabitEthernet4/0/1
 switchport access vlan 3
 switchport mode access
 power inline static
end

C9400-1#show power  inline upoe-plus gigabitEthernet 4/0/2
Codes: DS - Dual Signature device, SS - Single Signature device
       SP - Single Pairset device

Interface   Admin  Type Oper-State       Power(Watts)     Class   Device Name
            State       Alt-A,B       Allocated Utilized  Alt-A,B
----------- ------ ---- ------------- --------- --------- ------- -----------

Gi4/0/2     static SS   on,on         90.0      8.8       8       Ieee PD
C9400-1#
```

We can see that the admin state has changed from auto to static.

```
C9400-1#show power inline gigabitEthernet 4/0/2 detail
 Interface: Gi4/0/2
 Inline Power Mode: static
 Operational status (Alt-A,B): on,on
C9400-1#
```

Step 4: Now remove the static power by using "no power inline static" on the same interface and see the detailed output.

```
C9400-1(config)#
```

< Copy Paste in exec mode >

```
conf t
   interface GigabitEthernet4/0/2
      no power inline static
end
```

```
C9400-1#show power inline gigabitEthernet 4/0/2 detail
 Interface: Gi4/0/2
 Inline Power Mode: auto
 Operational status (Alt-A,B): on,on
C9400-1#
```

We can see that the admin status has been changed to auto from static and this CLI gives us much more information for the particular interface power consumption. This concludes our lab.

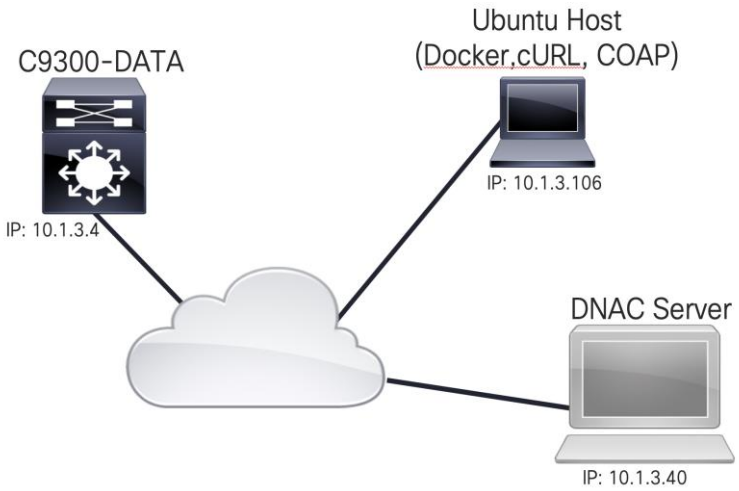
Scenario Configure App Hosting

Application Hosting on Cisco Catalyst 9000 Series switches opens up new innovation opportunities by converging network connectivity with a distributed application runtime environment, including hosting applications developed by partners and 3rd-party developers. Cisco IOx on a Catalyst 9K supports applications containerized in Linux KVM-based virtual Machines and Linux LXC containers. From 16.12.1, only native Docker is supported.

Hosted applications can be managed through Command Line Interface (CLI), and Cisco DNA Center, which will provide a centralized user interface to deploy and manage the entire lifecycle of the applications.

In this lab activity, you will learn how to install a 3rd party application via Cisco DNA Center.

Network Diagram App Hosting



Task 1: Configure App Hosting

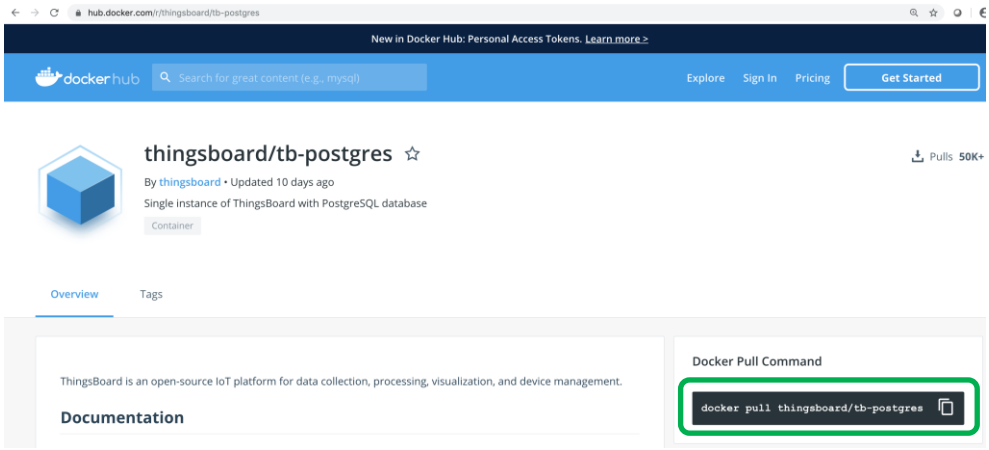
Step 1: Download the ThingsBoard application from Docker Hub.

In this lab, we will install an open source application called ThingsBoard.

Note: Thingsboard is an open-source IoT platform for data collection, processing, visualization, and device management. It enables device connectivity via industry standard IoT protocols - MQTT, CoAP and HTTP and supports both cloud and on-premises deployments. ThingsBoard combines scalability, fault-tolerance and performance so you will never lose your data.
We will start the lab as follow:

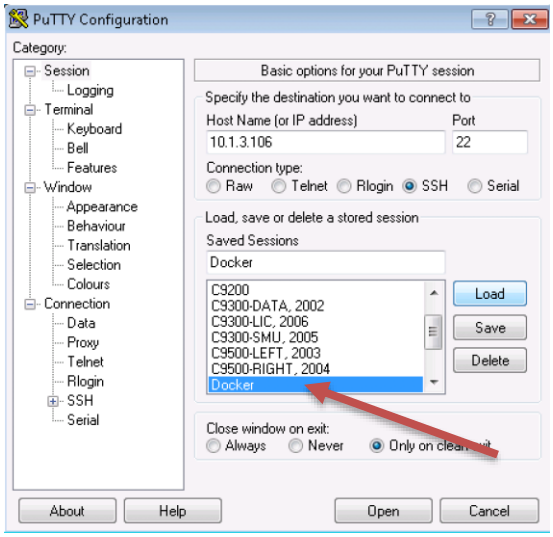
- Getting the application from Docker Hub and saving as ".tar" file. Note: (today the only supported format is .tar)
- Install the application to the C9300-DATA switch from Cisco DNA Center.
- Validate the the application by sending simulated IoT data using the COAP protocol.

First, Please click the following link and click the "Docker Pull Command."
https://hub.docker.com/r/thingsboard/tb-postgres



Note: Docker Hub is a service provided by Docker for finding and sharing container images.

Login to Docker from Putty Session.
Username : *cisco*
Password : *cisco*



Paste docker pull command we copied from Docker Hub after adding "*sudo*".

```
cisco@U-Srv-106:~$ sudo docker pull thingsboard/tb-postgres
Using default tag: latest
latest: Pulling from thingsboard/tb-postgres
c5e155d5a1d1: Pull complete
221d80d00ae9: Pull complete

Digest: sha256:eefaf1b4f17ec5358dd73cd6423454825b3b24ed22bbd930af7fc4898521de6a
Status: Downloaded newer image for thingsboard/tb-postgres:latest
```

Once it is downloaded, you can verify with "docker images" command.

```
cisco@U-Srv-106:~$ sudo docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
thingsboard/tb-postgres   latest              0fc48d973134        11 days ago         1.04GB
```

Now, we are ready to save this docker image as "tar" format to host on Catalyst 9300 DATA switch.

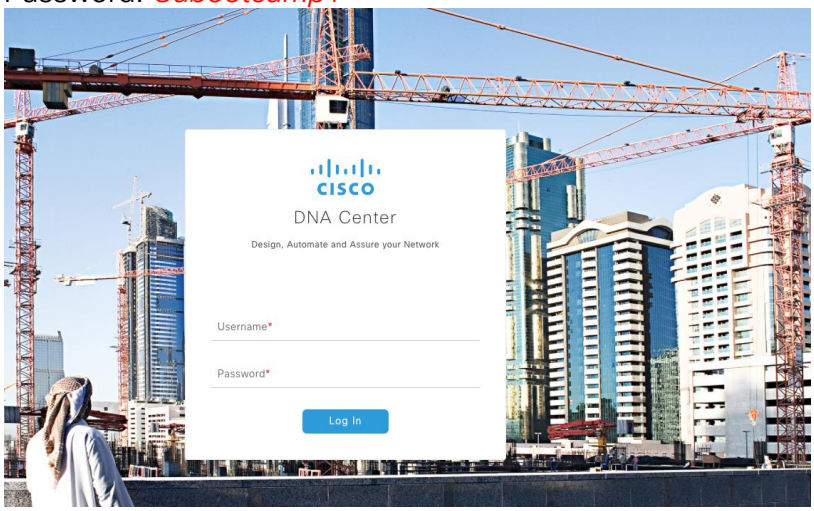
```
cisco@U-Srv-106:~$ sudo docker save thingsboard/tb-postgres -o tb.tar
```

```
cisco@U-Srv-106:~$ ls
bigmuddy-network-telemetry-pipeline  Downloads     metrics.json  Public        Templates
Desktop                              googlelogo_color_272x92dp.png  Music         simpleztp.py  Videos
Documents                            grpc                           Pictures      tb.tar
cisco@U-Srv-106:~$
```

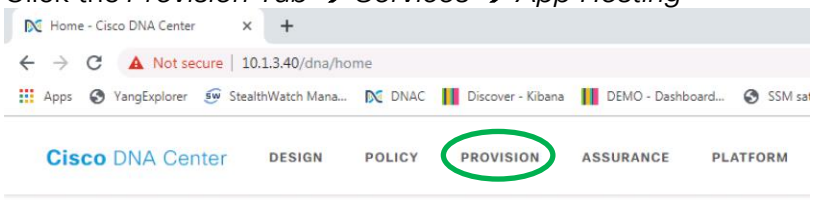
Note: We have copied this image to the Jump Host PC where we will deploy this app on the C9300 DATA switch from Cisco DNA Center.

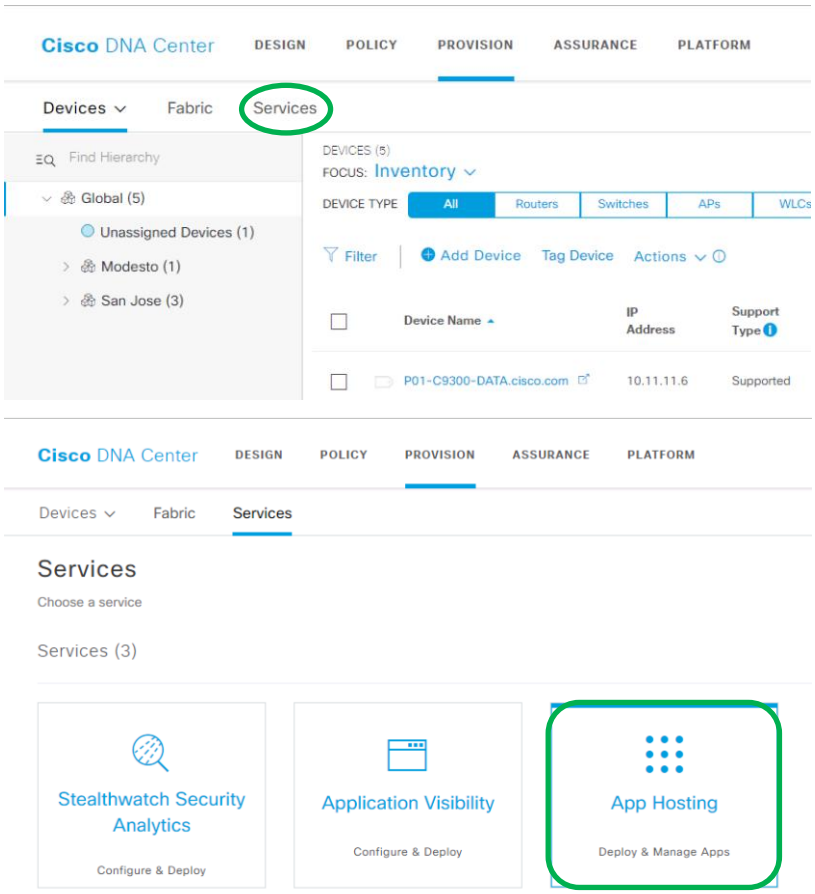
Step 2: Login to Cisco DNA-Center and upload the application.

Login to the Cisco DNA Center server *https://10.1.3.40* via Google Chrome.
Username: *admin*
Password: *Uabootcamp1*



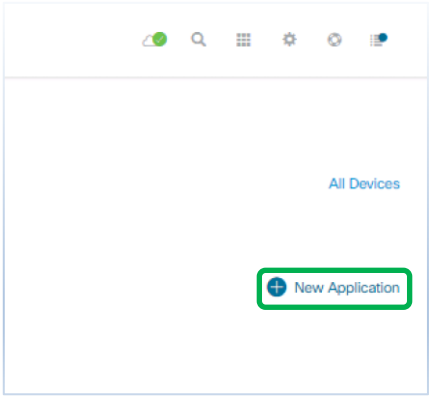
Click the *Provision Tab* → *Services* → *App Hosting*





Now you will see App Hosting page where you can manage all your applications.

Click "*New Application*" from the page.



Define "*Type*", "*Category*" and Click "*Select*" Button to upload the app.

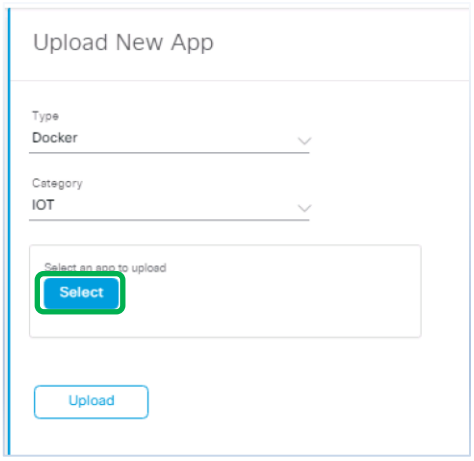
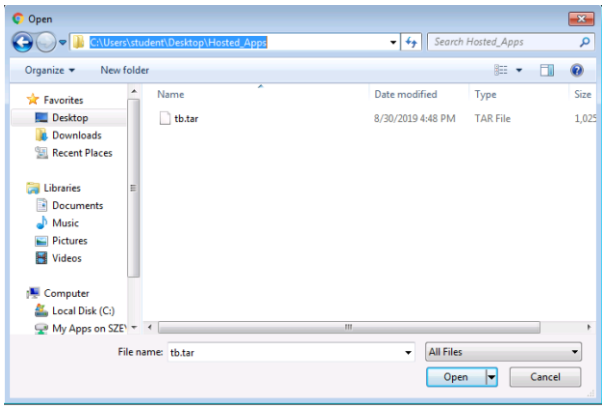
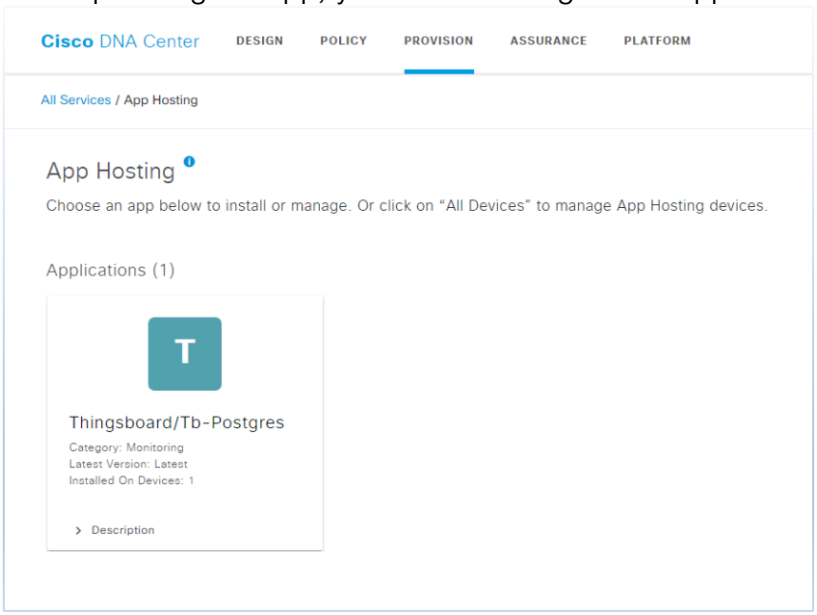


Image location is *Desktop* → *Hosted Apps* → *tb.tar*

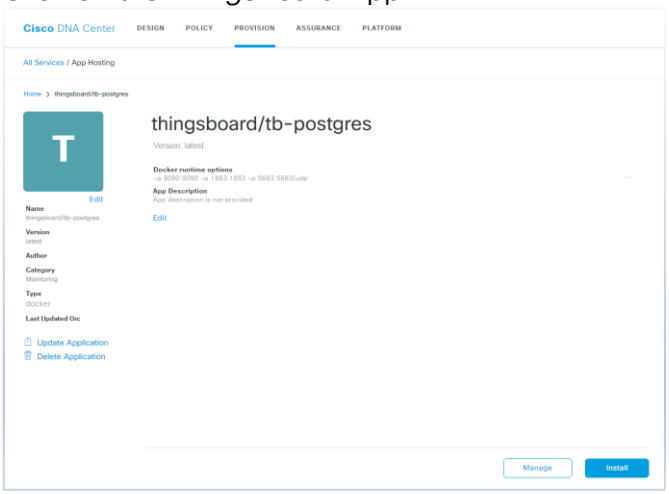


After uploading the app, you will see ThingsBoard application in the dashboard.



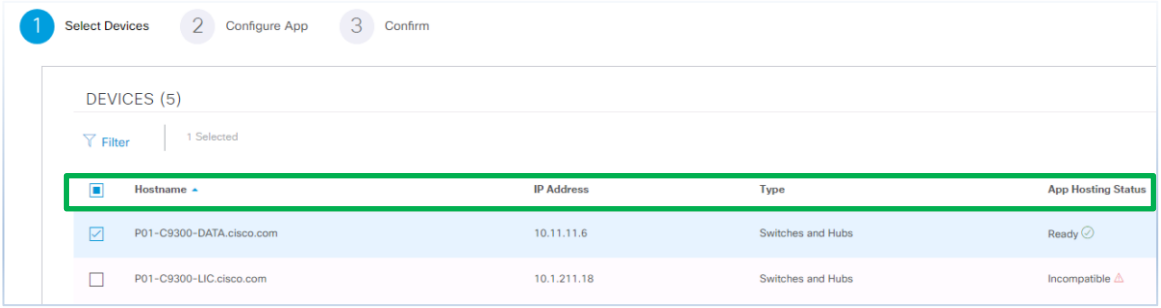
Step 3: Application Installation to C9300-DATA switch.

In this step, we will install the ThingsBoard app on the C9300-DATA switch. Click on the *ThingsBoard* App.

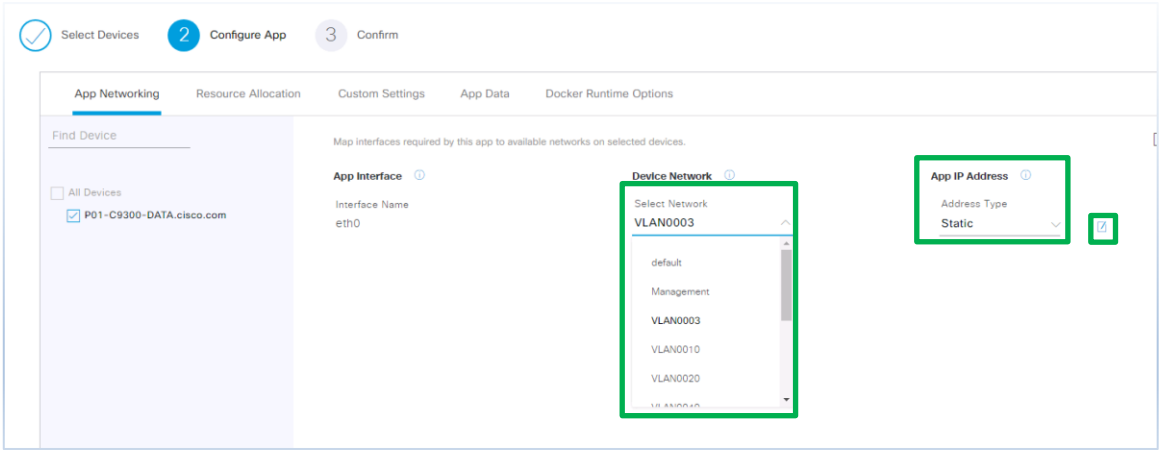


Then Click *Install*.

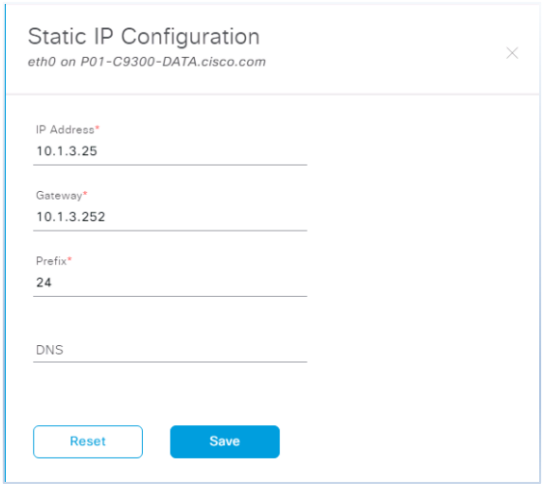
In this lab, we will select the *C9300-DATA* switch and Click *Next*.



In this step, we will choose *Vlan3* for the application network and select *Static* for IP address. Click on the small Box to add the static IP.

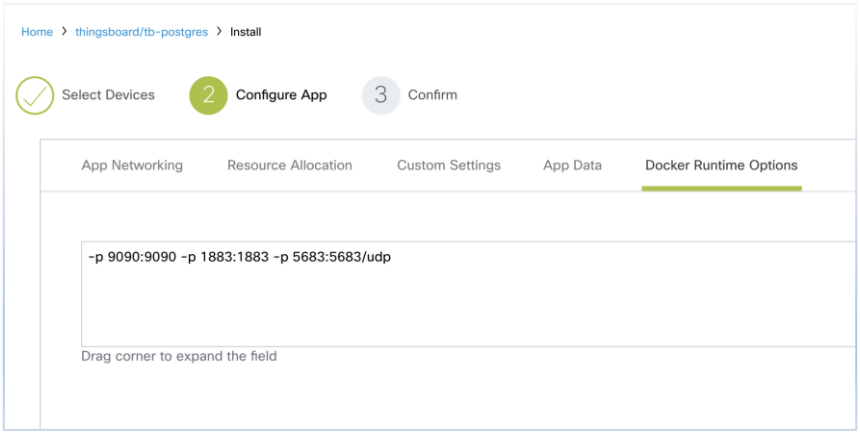


IP information for ThingsBoard App:

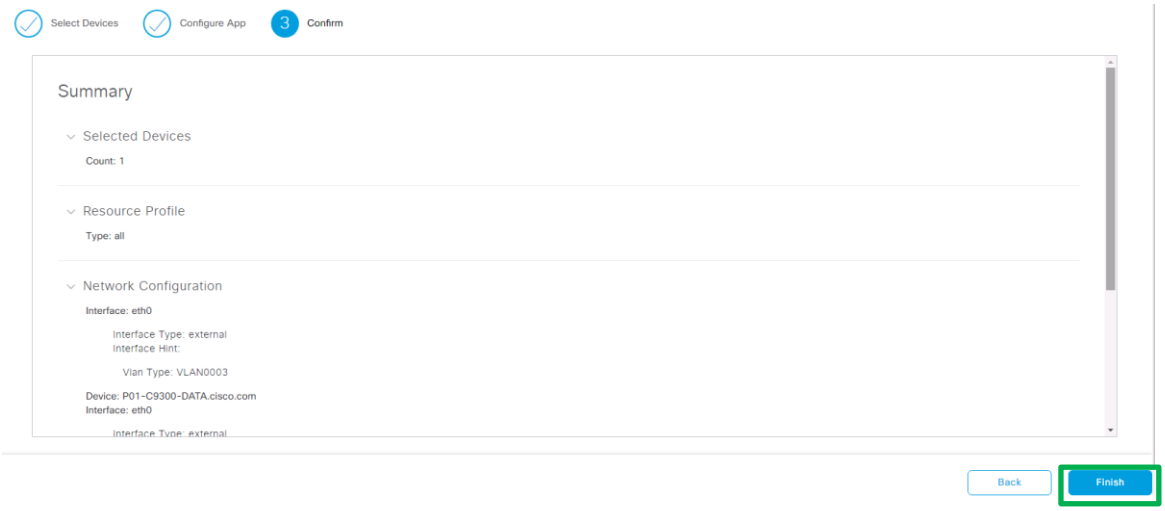


After Clicking "*Save*" , select the "*Docker Runtime Options*" tab and add " *-p 9090:9090 -p 1883:1883 -p 5683:5683/udp* " to expose internal ports to local ports.

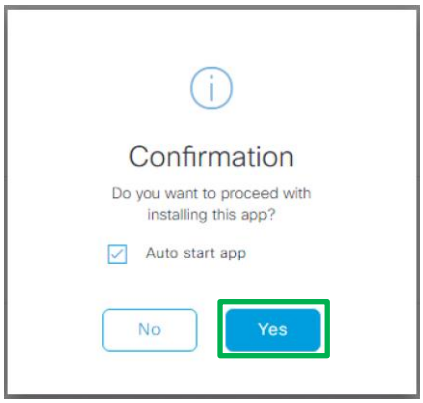
- -p 9090:9090 - connect local port 9090 to exposed internal HTTP port 9090
- -p 1883:1883 - connect local port 1883 to exposed internal MQTT port 1883
- -p 5683:5683 - connect local port 5683 to exposed internal COAP port 5683



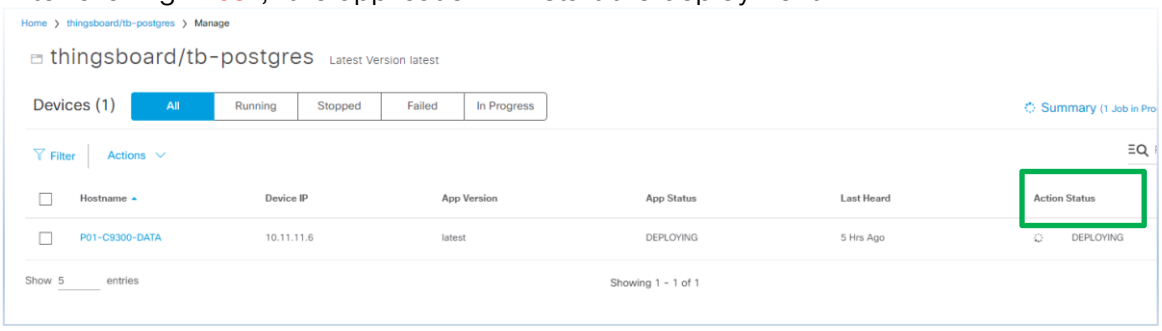
Click "*Next*" and then click "*Finish*".



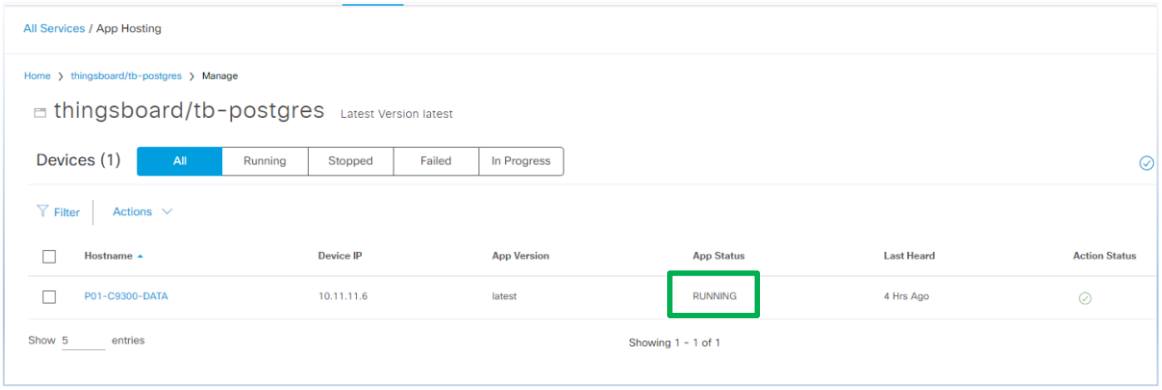
Review the configuration summary and click "*Finish*".



After clicking "*Yes*", the application will start the deployment.



Please wait till ThingsBoard is successfully installed and running. This process will take approximately 2 minutes.



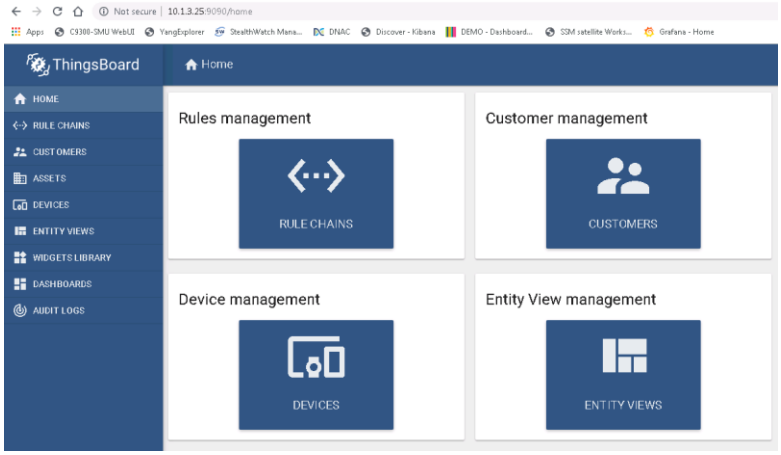
Task 2: Verify Installed Application

Step 1: Login to the ThingsBoard

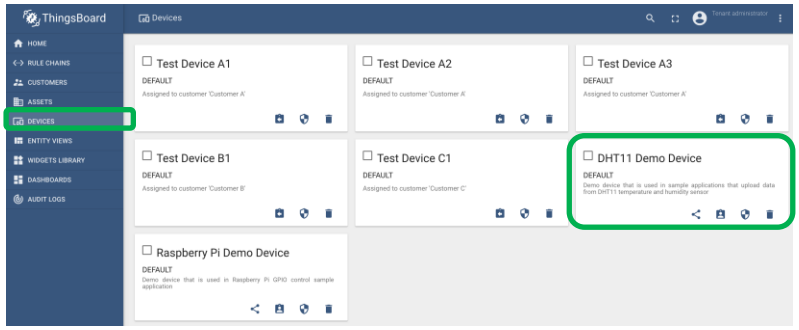
From the *Jump PC's Google Chrome*, please access to *http://10.1.3.25:9090* and you will see the dashboard of ThingsBoard.

Username: *tenant@thingsboard.org*

Password : *tenant*



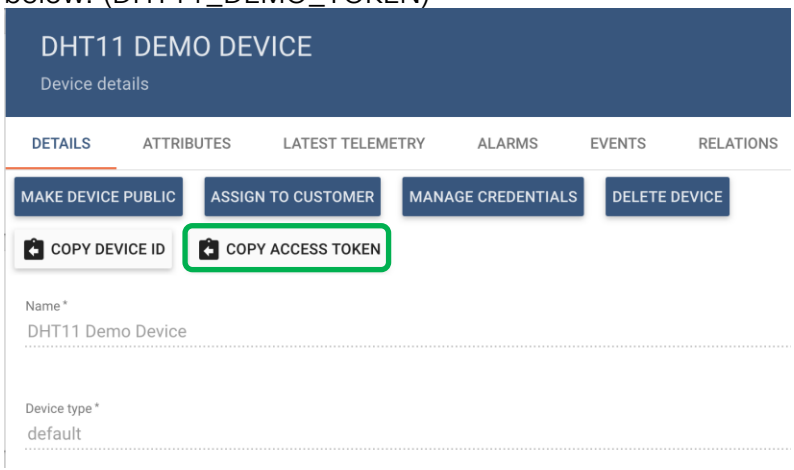
Click the "*Devices*" tab and then Select " *DHT11 Demo Device*"



This lab is based on an IoT device monitoring use case. We will show how to monitor temperature and humidity of the IOT device and visualize collected data.
To simplify the process, we will simulate the IoT data manually by pushing data using HTTP(cURL) and CoAP protocols from your lab pod.

HTTP(cURL) Format
curl -v -X POST -d "{"temperature": 21, "humidity": 55}" $HOST_NAME/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:application/json"

Note: Replace $HOST_NAME and $ACCESS_TOKEN with corresponding values.
$HOST_Name → http://10.1.3.25:9090
$ACCESS_TOKEN → Get this data by clicking "COPY ACCESS TOKEN" button as below. (DHT11_DEMO_TOKEN)

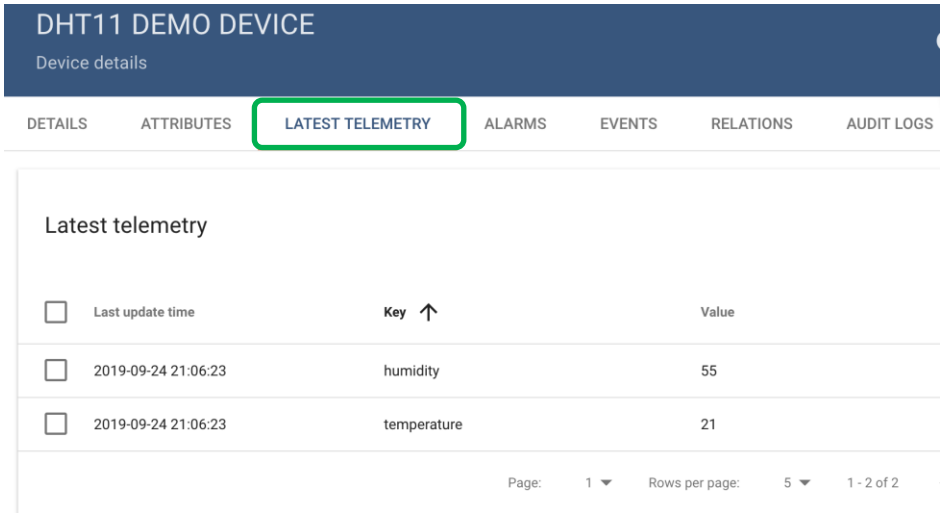


In this lab, we have already replaced this data for you.
Copy below cURL command and run this from Ubuntu Server we used for Docker image.

```
curl -v -X POST -d "{"temperature": 21, "humidity": 55}"
http://10.1.3.25:9090/api/v1/DHT11_DEMO_TOKEN/telemetry --header "Content-
Type:application/json"
```

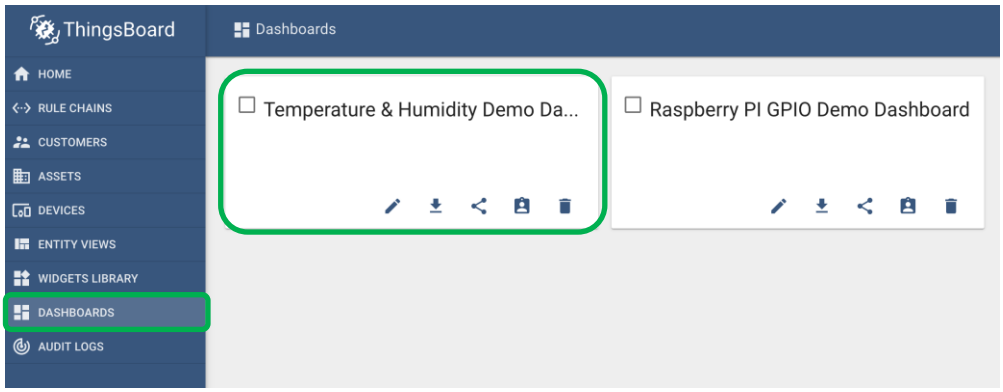
```
cisco@U-Srv-106:/$
cisco@U-Srv-106:/$ curl -v -X POST -d "{"temperature": 21, "humidity": 55}" http://10.1.3.25:9090/api/v1/DHT11_DEMO_TOKEN/tele
metry --header "Content-Type:application/json"
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 10.1.3.25...
* TCP_NODELAY set
* Connected to 10.1.3.25 (10.1.3.25) port 9090 (#0)
> POST /api/v1/DHT11_DEMO_TOKEN/telemetry HTTP/1.1
> Host: 10.1.3.25:9090
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Type:application/json
> Content-Length: 31
>
* upload completely sent off: 31 out of 31 bytes
< HTTP/1.1 200
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< Content-Length: 0
< Date: Wed, 25 Sep 2019 04:32:51 GMT
<
* Connection #0 to host 10.1.3.25 left intact
cisco@U-Srv-106:/$
```

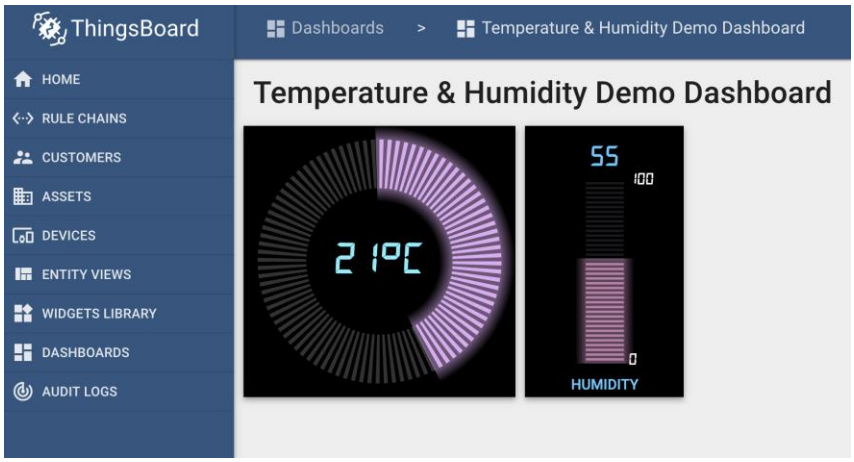
After you have pushed the data from this host server, Click "*LATEST TELEMETRY*". You will see telemetry data is published for this device at ThingsBoard page as below.



Now we will see this data from Dashboard. Click "*DASHBOARDS*" and Select "*Temperature & Humidity Demo …*".

Note: It may take up to 15 minutes the dashboard to start as it need to collect data.





COAP Format

cat telemetry-data.json | coap post coap://$THINGSBOARD_HOST/api/v1/$ACCESS_TOKEN/telemetry

Note:

telemetry-data.json → {"temperature":81, "humidity":34}

$THINGSBOARD_HOST → 10.1.3.25:5683 (5683 is port for COAP)

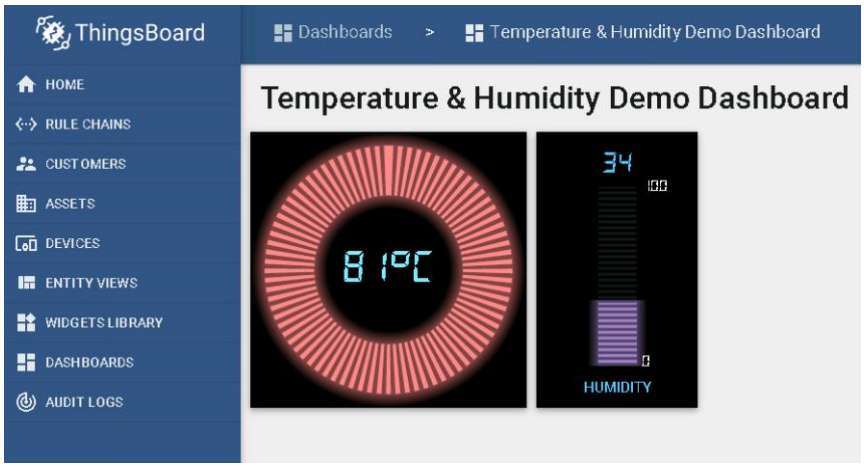
$ACCESS_TOKEN → DHT11_DEMO_TOKEN

Please run below COAP command to push data to ThingsBoard.

cat telemetry-data.json | coap post
coap://10.1.3.25:5683/api/v1/DHT11_DEMO_TOKEN/telemetry

```
cisco@U-Srv-106:~$ cat telemetry-data.json | coap post coap://10.1.3.25:5683/api/v1/DHT11_DEMO_TOKEN/telemetry
(2.03)
cisco@U-Srv-106:~$
```

Please refresh the "*Dashboards*" page and you will see new telemetry data in application dashboard.



We have successfully tested the application using cURL and COAP format.

Scenario Demonstrate xFSU on Catalyst 9300 Stack

When a normal software upgrade is executed, the control plane and data plane are reset at the same time. This result in an impact for all the user traffic. With Extended Fast Software Upgrade (xFSU), the control plane and data plane update is segregated. The control plane gets upgraded first while the traffic is flowing through the switch or switch stack, and only then the data plane is reset with a special mechanism called *Cache and Flush*. The system caches the current forwarding entries in a special memory block before the reset, and it is then flushed / reprogrammed in the ASIC after the reset causing data traffic impact of less than 30 seconds.

We can also use the *Fast Reload* feature to reload the switch, which behaves the same way as the Extended Fast Software Upgrade (xFSU) but without upgrading the image on the switch.

xFSU commands:

1. Use “reload fast “ command instead of “reload” command to invoke xFSU.

2. For upgrading to a new image using xFSU, use the following CLI:
   “install add file flash://<path-image> activate reloadfast commit”

Prerequisites

Before starting an Extended Fast Software Upgrade, ensure that the image on your Catalyst 9300 switch stack is Cisco IOS XE Release 17.1.1 or later and is running in install mode.

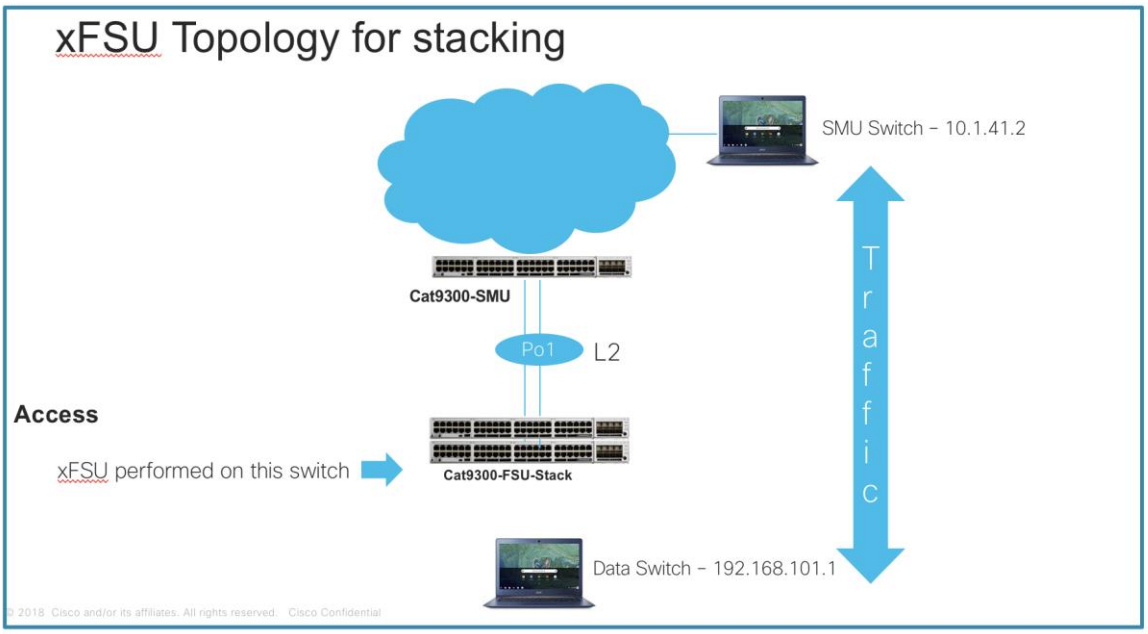
Restrictions:

- Extended Fast Software Upgrade (xFSU) is supported only on access switches with the following conditions:
  - The switch undergoing (xFSU) should not be Spanning tree root
  - The switch undergoing (xFSU) should not have more than one forwarding port for the same Vlan

In this lab exercise, you will learn how to use xFSU utility in two ways:

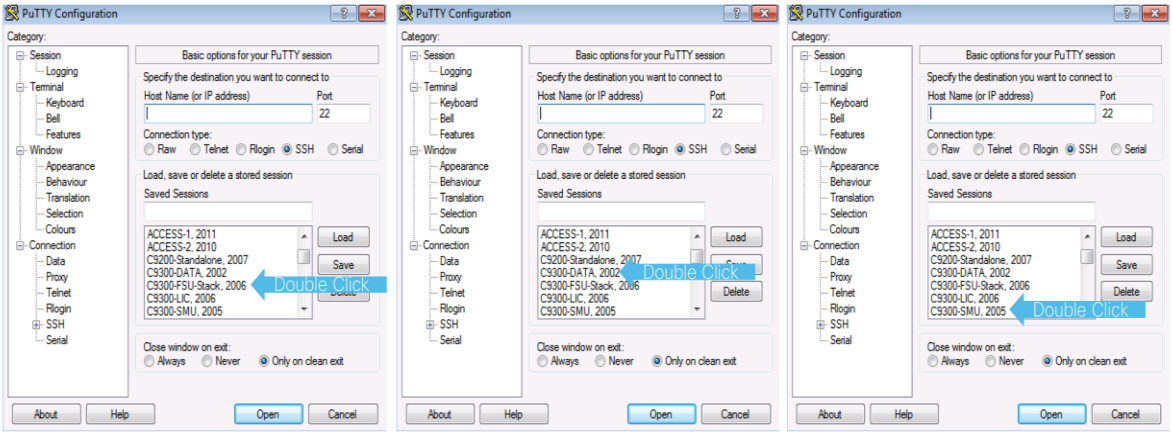
1. Reload Fast – The switch stack will go through reload only leveraging the xFSU software architecture resulting in traffic loss of less than 30 seconds.
2. Software Upgrade – The switch stack will be upgraded with new software leveraging the xFSU architecture also resulting in traffic loss of less than 30 seconds.

Network Diagram xFSU



Task 1: Obverse traffic downtime during "reload fast" operation.

Connect to CC9300-FSU-Stack, Data and SMU as shown below:



Step 1: Verify the xFSU requirement.

Verify if you are running the desired images on the stack in *Install mode* as shown below. In this lab, we are using engineering beta release.

You will see that current running image is in *INSTALL mode*.

```
C9300-FSU-Stack# show ver | beg SW Version

Switch Ports Model              SW Version        SW Image              Mode
------ ----- -----              ----------        ----------            ----
*    1 41    C9300-24U          17.1.1            CAT9K_IOSXE           INSTAL
     2 41    C9300-24T          17.1.1            CAT9K_IOSXE           INSTALL
```

Note: Please check if the STP conditions are met by following the below steps:

```
C9300-FSU-Stack# show spanning-tree summary
Switch is in rapid-pvst mode
Root bridge for: none
EtherChannel misconfig guard            is enabled
Extended system ID                      is enabled
Portfast Default                        is disabled
PortFast BPDU Guard Default             is disabled
Portfast BPDU Filter Default            is disabled
Loopguard Default                       is disabled
UplinkFast                               is disabled
BackboneFast                             is disabled
Configured Pathcost method used is short
Name                   Blocking  Listening  Learning  Forwarding  STP Active
---------------------- --------  ---------  --------  ----------  ----------
VLAN0001                      0          0         0           7           7
VLAN0003                      0          0         0           1           1
VLAN0041                      0          0         0           1           1
VLAN0211                      0          0         0           1           1
---------------------- --------  ---------  --------  ----------  ----------
4 vlans                       0          0         0          10          10
```

Note: If you observe that the switch is root for some vlans then xFSU will not work. Please add the below configuration and then proceed with next steps:

```
C9300-FSU-Stack(config)# spanning-tree vlan 3,41 priority 61440
C9300-FSU-Stack(config)# no vlan 192
C9300-FSU-Stack(config)# no vlan 211
```

Note: xFSU will work on 9300 Switch only if the above STP criteria is satisfied. If the 9300 switch is deployed in access layer, then you will not run in to the STP issues but if they are deployed in Distribution/Core layer then you will have to perform necessary changes to satisfy the STP criteria

Note: xFSU will only check the STP table for non-edge ports. For all access ports, it is mandatory to configure port-fast feature so that it does not participate in STP Learning.

Step 2: Initiate ping traffic from Data to SMU Switch as shown in above picture and then perform reload fast operation on the stack

The main goal of initiating ping is to have some traffic flowing through the switch stack when reload fast is performed so that traffic loss can be measured appropriately.

First, check if you have appropriate OSPF routes for the destination switch and if the Graceful Reload Infra is in the desired state. Graceful Reload Infrastructure in IOS-XE makes sure that the protocols do not go down during the reload or software upgrade.

```
C9300-DATA# show ip route 10.1.41.2
Routing entry for 10.1.41.0/24
  Known via "ospf 1", distance 110, metric 2, type intra area
  Last update from 192.168.102.2 on Port-channel1, 2d21h ago
  Routing Descriptor Blocks:
  * 192.168.102.2, from 192.168.102.2, 2d21h ago, via Port-channel1
      Route metric is 2, traffic share count is 1

C9300-DATA# ping 10.1.41.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.41.2, timeout is 2 seconds:
.!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

Graceful Reload Infra should be in " *Not running* " state before performing reload fast operation.

```
C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: Not running
Minimum required system uptime before fast reload can be supported is 5
seconds
Client OSPFV3                              : (0x10203005) Status: Up
Client OSPF                                : (0x10203004) Status: Up
Client GR_CLIENT_BGP                       : (0x10203003) Status: Up
Client IS-IS                               : (0x10203002) Status: Up
Client GR_CLIENT_RIB                       : (0x10203001) Status: Up
Client GR_CLIENT_FIB                       : (0x10203000) Status: Up
```

If the above steps are working as expected, run a continuous ping from the Data to SMU Switch as shown below before starting the reload fast operation.

```
C9300-DATA# ping 10.1.41.2 repeat 1000000 timeout 1
```

Step 3: Start Fast Reload

Now, let's start the test of reload fast on the Stack Switch. You will observe the following events during the reload:

1. With Active switch intact, Standby switch goes for a reload and comes back online to form SSO with current Active Switch
2. Active Switch goes for a reload triggering auto-switchover causing the standby switch to become the new active
3. Active Switch comes back online to become the new standby and forms SSO with new Active Switch

```
C9300-FSU-Stack# write mem
C9300-FSU-Stack# reload fast
Reload fast command is being issued on Active unit, this will reload fast
the whole stack
Proceed with reload fast? [confirm]
```

```
Checking STP eligibility: Eligible SUCESS: Fast reload requirement pre-check
--- Verifying Platform specific xFSU admission criteria --- SUCESS: Fast reload image pre-
check
Check fast reload support and verification on switch
[2]: fast_rld_verify package(s) on switch 2
Finished preverifying before fast reload

SUCCESS to verify packages

SUCCESS to verify before fast reload

[2]: Finished fast_rld_verify successful on switch 2
   (-2) SUCCESS: Finished fast_rld_verify: Success on [2]
[1 2]: Performing Upgrade_Service

*Oct 01 10:36:54.470: %IOSXEBOOT-4-BOOTLOADER_UPGRADE: (local/local): Starting boot
preupgrade
300+0 records in
300+0 records out
307200 bytes (307 kB, 300 KiB) copied, 0.31446 s, 977 kB/s
mount: /tmp/microcode_update/boot_pkg: WARNING: device write-protected, mounted read-only.
  SUCCESS: Upgrade_Service finished

========================================
Stage 1/2: Fast reloading Standby/Members
========================================
   (-2) --- Starting wait for Standby to reach terminal redundancy state ---

000180: *Oct  1 10:36:57.059: Locking Config during Fast Reload operation
000181: *Oct  1 10:36:57.488: %STACKMGR-1-RELOAD: Switch 2 R0/0: stack_mgr: Reloading due to
reason Standby & Members fast reload command
000182: *Oct  1 10:37:12.516: %HMANRP-5-CHASSIS_DOWN_EVENT: Chassis 2 gone DOWN!
000183: *Oct  1 10:37:12.589: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_NOT_PRESENT)
000184: *Oct  1 10:37:12.589: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault (PEER_DOWN)
000185: *Oct  1 10:37:12.589: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_REDUNDANCY_STATE_CHANGE)
000186: *Oct  1 10:37:12.604: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 removed
000187: *Oct  1 10:37:12.739: %RF-5-RF_RELOAD: Peer reload. Reason: EHSA standby down
000188: *Oct  1 10:37:12.754: %IOSXE_REDUNDANCY-6-PEER_LOST: Active detected switch 2 is no
longer standby
000189: *Oct  1 10:37:12.524: %STACKMGR-5-SWITCH_REMOVED: Switch 1 R0/0: stack_mgr: Switch 2
has been removed from the stack.
000190: *Oct  1 10:40:00.085: %STACKMGR-5-SWITCH_ADDED: Switch 1 R0/0: stack_mgr: Switch 2
has been added to the stack.
000191: *Oct  1 10:40:01.523: %PLATFORM_FEP-6-FRU_PS_OIR: Switch 2: FRU power supply A
inserted
000192: *Oct  1 10:40:01.523: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 1
inserted
000193: *Oct  1 10:40:01.524: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 2
inserted
000194: *Oct  1 10:40:01.524: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 3
inserted
000195: *Oct  1 10:40:02.028: %STACKMGR-5-SWITCH_ADDED: Switch 1 R0/0: stack_mgr: Switch 2
has been added to the stack.
000196: *Oct  1 10:40:04.091: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for
switch 2: EMP_RELAY: Channel DOWN!
000197: *Oct  1 10:40:04.704: %PLATFORM_FEP-6-FRU_PS_OIR: Switch 2: FRU power supply A
inserted
000198: *Oct  1 10:40:04.739: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000199: *Oct  1 10:40:04.740: %ENT_API-4-NOPARENT: Parent physical entity 2000 did not exist
when trying to add
         child physical entity 2030, phyDescr = usbflash1-2, phyName = usbflash1-2.
000200: *Oct  1 10:40:04.822: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000201: *Oct  1 10:38:57.331: %IOSXE-0-PLATFORM: Switch 2 R0/0: udev: usb1: has been inserted
000202: *Oct  1 10:39:58.284: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack
port 1 on Switch 2 is down
000203: *Oct  1 10:39:58.721: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack
port 1 on Switch 2 is up
000204: *Oct  1 10:39:58.722: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack
port 2 on Switch 2 is up
```

```
000205: *Oct  1 10:39:59.288: %STACKMGR-5-SWITCH_ADDED: Switch 2 R0/0: stack_mgr: Switch 2
has been added to the stack.
000206: *Oct  1 10:40:01.257: %STACKMGR-5-SWITCH_ADDED: Switch 2 R0/0: stack_mgr: Switch 2
has been added to the stack.
000207: *Oct  1 10:40:07.489: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000208: *Oct  1 10:40:08.385: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for
switch 2: EMP_RELAY: Channel UP!
000209: *Oct  1 10:42:04.717: %IOSXE_REDUNDANCY-6-PEER: Active detected switch 2 as standby.
000210: *Oct  1 10:42:04.710: %STACKMGR-6-STANDBY_ELECTED: Switch 1 R0/0: stack_mgr: Switch 2
has been elected STANDBY.
000211: *Oct  1 10:42:09.788: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby
insertion (raw-event=PEER_FOUND(4))

000212: *Oct  1 10:42:09.789: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby
insertion (raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))

000213: *Oct  1 10:43:06.969: %HA_CONFIG_SYNC-6-BULK_CFGSYNC_SUCCEED: Bulk Sync succeeded
000214: *Oct  1 10:43:07.972: %RF-5-RF_TERMINAL_STATE: Terminal state reached for
(SSO)Standby has reached SSO hot
==========================================
Stage 2/2: Fast reloading Active(Switchover)
==========================================
Check fast reload support and verification on switch

000215: *Oct  1 10:43:17.276: %FED_IPC_MSG-5-FAST_RELOAD_COMPLETE: Switch 2 R0/0: fed: Fast
reload operation complete[1]: fast_rld_verify package(s) on switch 1
Finished preverifying before fast reload
SUCCESS to verify packages
SUCCESS to verify before fast reload
[1]: Finished fast_rld_verify successful on switch 1
   (-2) SUCCESS: Finished fast_rld_verify: Success on [1]

Extracting /tmp/vmlinux from /flash/cat9k-
rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
6422576+0 records in
6422576+0 records out
6422576 bytes (6.4 MB, 6.1 MiB) copied, 15.7687 s, 407 kB/s
Extracting /tmp/initramfs.cpio.gz from /flash/cat9k-
rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg

000216: *Oct  1 10:45:04.121: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for
switch 2: EMP_RELAY: Channel DOWN!
000217: *Oct  1 10:45:05.775: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for
switch 2: EMP_RELAY: Channel UP!38042177+0 records in
38042177+0 records out
38042177 bytes (38 MB, 36 MiB) copied, 94.4628 s, 403 kB/s
/usr/sbin/kexec -l /tmp/vmlinux --append="root=/dev/ram rw console=tty0,9600n8 max_loop=64
pciehp.pciehp_force unknown_nmi_panic BOARDID=20566 bdinfo_start=0xAC9CA018
bdinfo_size=0x3AC70 rd_start=0x8BCF7000 rd_size=0x2447DBF pkg_start=0x0 pkg_size=0x0
fastreload 1ru stack SR_BOOT=flash:packages.conf  intel_pstate=disable  intel_pstate=disable
intel_pstate=disable" --ramdisk=/tmp/initramfs.cpio.gz
/bin/systemctl start kexec.target
Oct098: %PMAN-5-EXITACTION: F0/0: pvp: Process manager is exiting:
Oct  1 10:45:16.988: %PMAN-5-EXITACTION: R0/0: pvp: Process manager is exiting:

000180: *Oct  1 10:48:35.511: %STACKMGR-5-FAST_RELOAD_DONE: Switch 2 R0/0:
stack_mgr: Fast reload operation completed successfully on entire stack
000181: *Oct  1 10:48:36.357: %FED_IPC_MSG-5-FAST_RELOAD_COMPLETE: Switch 1 R0/0:
fed: Fast reload operation complete
```

Once the switch comes back online with prompt; please check the output of graceful reload infrastructure using the below command. The protocol status should be in "*Up*" status as shown. *It can generally take about 4-5 minutes.*

```
C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: Dataplane update granted
Minimum required system uptime before fast reload can be supported is 5 seconds
Client OSPFV3                          : (0x10203005) Status: Dataplane update granted
```

```
Client OSPF                             : (0x10203004) Status: Dataplane update granted
Client GR_CLIENT_BGP                    : (0x10203003) Status: Dataplane update granted
Client IS-IS                            : (0x10203002) Status: Dataplane update granted
Client GR_CLIENT_RIB                    : (0x10203001) Status: Dataplane update granted
Client GR_CLIENT_FIB                    : (0x10203000) Status: Dataplane update granted

C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: Not running
Mnimum required system uptime before fast reload can be supported is 5 seconds
Client OSPFV3                           : (0x10203005) Status: Up
Client OSPF                             : (0x10203004) Status: Up
Client GR_CLIENT_BGP                    : (0x10203003) Status: Up
Client IS-IS                            : (0x10203002) Status: Up
Client GR_CLIENT_RIB                    : (0x10203001) Status: Up
Client GR_CLIENT_FIB                    : (0x10203000) Status: Up
```

Then, stop the pings by hitting *CTRL-Shift-6* if the ping is still going on and observe the traffic lost. It should be less than 30 seconds.

Step 4: You can also verify and check the reload reason after the completion of reload fast operation

```
C9300-FSU-Stack# show version
Cisco IOS XE Software, Version BLD_V171_THROTTLE_LATEST_20190906_004026_2
Cisco IOS Software [Amsterdam], Catalyst L3 Switch Software (CAT9K_IOSXE),
Experimental Version 17.1.20190906:011308 [v171_throttle-
/nobackup/mcpre/BLD-BLD_V171_THROTTLE_LATEST_20190906_004026 187]
Copyright (c) 1986-2019 by Cisco Systems, Inc.
Compiled Fri 06-Sep-19 00:39 by mcpre

Cisco IOS-XE software, Copyright (c) 2005-2019 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.

ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.1.1r, RELEASE SOFTWARE (P)

C9300-FSU-Stack uptime is 13 hours, 59 minutes
Uptime for this control processor is 17 minutes
System returned to ROM by SSO Switchover
System image file is "flash:packages.conf"
Last reload reason: Reload Fast Command
```

Task 2: Perform Extended Fast Software Upgrade(xFSU).

In this task, we will obverse traffic downtime with xFSU feature by upgrading to new software image. With xFSU, traffic downtime should be less than 30 seconds.

Step 1: Make sure you have desired images in the flash as shown below:

```
C9300-FSU-Stack# dir flash: | i .bin
459172  -rw-        719234956  Jan 23 2019 02:02:14 +00:00  cat9k-eFSU-1.bin
459137  -rw-        719230258  Jan 23 2019 04:07:38 +00:00  cat9k-eFSU-2.bin
```

If *any of the image* are not in flash, please copy as below:

```
C9300-FSU-Stack# copy flash:/FSU/cat9k-eFSU-2.bin flash:
Destination filename [cat9k-eFSU-2.bin]?
Copy in
progress...CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Step 2: Initiate a continuous ping traffic from Data Switch to SMU Switch flowing through the stack for measuring traffic convergence during the software upgrade.

```
C9300-DATA# ping 10.1.41.2 repeat 1000000 timeout 1
```

Step 3: Start the software upgrade using the Extended Fast Software Upgrade Utility(xFSU) after a quick verification.

Please check the software build version on the stack switch before proceeding with software upgrade

```
###################################

C9300-FSU-Stack# show version
Cisco IOS XE Software, Version BLD_V171_THROTTLE_LATEST_20190906_004026_2

if you see the above output with _2 in the end, you should upgrade to cat9k-eFSU-1.bin
image.

###################################

C9300-FSU-Stack# show version

Cisco IOS XE Software, Version BLD_V171_THROTTLE_LATEST_20190906_004026
if you see the above output without _2 in the end, you can proceed upgrading to cat9k-
eFSU-2.bin image.
```

Now, you can start the software upgrade to the desired image based on results from above verification:

```
C9300-FSU-Stack# install add file flash:cat9k-eFSU-2.bin activate reloadfast commit
install_add_activate_commit: START Fri Sep 20 03:58:48 UTC 2019
System configuration has been modified.
Press Yes(y) to save the configuration and proceed.
Press No(n) for proceeding without saving the configuration.
```

```
Press Quit(q) to exit, you may save configuration and re-enter the command. [y/n/q]y
Modified configuration has been saved
Checking STP eligibility: Eligible SUCESS: Fast reload requirement pre-check
*Sep 20 03:59:03.556: %INSTALL-5-INSTALL_START_INFO: Switch 2 R0/0: install_engine: Started
install one-shot flash:cat9k-eFSU-1.bininstall_add_activate_commit: Adding PACKAGE
install_add_activate_commit: Checking whether new add is allowed ....
install_add_activate_commit: Checking whether new add is allowed ....
--- Starting initial file syncing ---
[1]: Copying flash:cat9k-eFSU-2.bin from switch 1 to switch 2
[2]: Finished copying to switch 2
Info: Finished copying flash:cat9k-eFSU-2.bin to the selected switch(es)
Finished initial file syncing
--- Starting Add ---
Performing Add on all members
  [1] Add package(s) on switch 1
  [1] Finished Add on switch 1
  [2] Add package(s) on switch 2
  [2] Finished Add on switch 2
Checking status of Add on [1 2]
Add: Passed on [1 2]
Finished Add
Image added. Version: 17.1.1.0.274
install_add_activate_commit: Activating PACKAGE
Following packages shall be activated:
/flash/cat9k-wlc.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-webui.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-sipspa.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-sipbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-rpbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-guestshell.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-espbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
/flash/cat9k-cc_srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
--- Verifying Platform specific xFSU admission criteria --- SUCESS: Fast reload image pre-
check
This operation requires a fast reload of the system. Do you want to proceed? [y/n]y
--- Starting Activate ---
Performing Activate on all members
000267: *Sep 30 21:48:17.825: %INSTALL-5-INSTALL_AUTO_ABORT_TIMER_PROGRESS: Switch 1 R0/0:
rollback_timer: Install auto abort timer will expire in 7200 seconds
[1] Activate package(s) on switch 1
    --- Starting list of software package changes ---
    Old files list:
      Removed cat9k-cc_srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-espbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-guestshell.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-rpbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-sipbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-sipspa.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-webui.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-wlc.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
    New files list:
      Added cat9k-cc_srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-espbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-guestshell.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-rpbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-sipbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-sipspa.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-webui.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-wlc.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
    Finished list of software package changes
  [1] Finished Activate on switch 1
  [2] Activate package(s) on switch 2
    --- Starting list of software package changes ---
```

```
    Old files list:
      Removed cat9k-cc_srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-espbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-guestshell.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-rpbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-sipbase.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-sipspa.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-webui.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg
      Removed cat9k-wlc.BLD_V171_THROTTLE_LATEST_20190906_004026.SSA.pkg

    New files list:
      Added cat9k-cc_srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-espbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-guestshell.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-rpbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-sipbase.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-sipspa.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-srdriver.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-webui.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
      Added cat9k-wlc.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg

    Finished list of software package changes
  [2] Finished Activate on switch 2

Checking status of Activate on [1 2]
Activate: Passed on [1 2]
Finished Activate
000268: *Sep 30 21:48:19.102: %INSTALL-5-INSTALL_AUTO_ABORT_TIMER_PROGRESS: Switch 2 R0/0:
rollback_timer: Install auto abort timer will expire in 7200 seconds--- Starting Commit ---
Performing Commit on all members
  [1] Commit package(s) on switch 1
  [1] Finished Commit on switch 1
  [2] Commit package(s) on switch 2
  [2] Finished Commit on switch 2
Checking status of Commit on [1 2]
Commit: Passed on [1 2]
Finished Commit
Check fast reload support and verification on switch
[2]: fast_rld_verify package(s) on switch 2
Finished preverifying before fast reload

SUCCESS to verify packages
SUCCESS to verify before fast reload
[2]: Finished fast_rld_verify successful on switch 2
   (-2) SUCCESS: Finished fast_rld_verify: Success on [2]
[1 2]: Performing Upgrade_Service
*Sep 30 21:49:13.046: %IOSXEBOOT-4-BOOTLOADER_UPGRADE: (local/local): Starting boot
preupgrade
300+0 records in
300+0 records out
307200 bytes (307 kB, 300 KiB) copied, 0.313938 s, 979 kB/s
  SUCCESS: Upgrade_Service finished
==========================================
Stage 1/2: Fast reloading Standby/Members
==========================================
   (-2) --- Starting wait for Standby to reach terminal redundancy state ---
000269: *Sep 30 21:49:16.143: Locking Config during Fast Reload operation
000270: *Sep 30 21:49:16.576: %STACKMGR-1-RELOAD: Switch 2 R0/0: stack_mgr: Reloading due to reason
Standby & Members fast reload command
000271: *Sep 30 21:49:29.438: %HMANRP-5-CHASSIS_DOWN_EVENT: Chassis 2 gone DOWN!
000272: *Sep 30 21:49:29.527: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 removed
000273: *Sep 30 21:49:29.535: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_NOT_PRESENT)
000274: *Sep 30 21:49:29.542: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault (PEER_DOWN)
000275: *Sep 30 21:49:29.542: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_REDUNDANCY_STATE_CHANGE)
000276: *Sep 30 21:49:29.696: %RF-5-RF_RELOAD: Peer reload. Reason: EHSA standby down
000277: *Sep 30 21:49:29.708: %IOSXE_REDUNDANCY-6-PEER_LOST: Active detected switch 2 is no longer
standby
```

```
000278: *Sep 30 21:49:29.444: %STACKMGR-5-SWITCH_REMOVED: Switch 1 R0/0: stack_mgr: Switch 2 has
been removed from the stack.
000279: *Sep 30 21:52:19.857: %STACKMGR-5-SWITCH_ADDED: Switch 1 R0/0: stack_mgr: Switch 2 has been
added to the stack.
000280: *Sep 30 21:52:21.115: %PLATFORM_FEP-6-FRU_PS_OIR: Switch 2: FRU power supply A inserted
000281: *Sep 30 21:52:21.116: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 1 inserted
000282: *Sep 30 21:52:21.116: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 2 inserted
000283: *Sep 30 21:52:21.117: %PLATFORM_THERMAL-6-FRU_FAN_OIR: Switch 2: System fan 3 inserted
000284: *Sep 30 21:52:21.837: %STACKMGR-5-SWITCH_ADDED: Switch 1 R0/0: stack_mgr: Switch 2 has been
added to the stack.
000285: *Sep 30 21:52:23.819: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch 2:
EMP_RELAY: Channel DOWN!
000286: *Sep 30 21:52:24.561: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000287: *Sep 30 21:52:24.562: %ENT_API-4-NOPARENT: Parent physical entity 2000 did not exist when
trying to add child physical entity 2030, phyDescr = usbflash1-2, phyName = usbflash1-2.
000288: *Sep 30 21:52:24.611: %PLATFORM_FEP-6-FRU_PS_OIR: Switch 2: FRU power supply A inserted
000289: *Sep 30 21:52:24.647: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000290: *Sep 30 21:52:24.648: %ENT_API-4-NOPARENT: Parent physical entity 2000 did not exist when
trying to add child physical entity 2030, phyDescr = usbflash1-2, phyName = usbflash1-2.
000291: *Sep 30 21:51:16.457: %IOSXE-0-PLATFORM: Switch 2 R0/0: udev: usb1: has been inserted
000292: *Sep 30 21:52:18.444: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack port 1
on Switch 2 is down
000293: *Sep 30 21:52:18.909: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack port 1
on Switch 2 is up
000294: *Sep 30 21:52:18.909: %STACKMGR-6-STACK_LINK_CHANGE: Switch 2 R0/0: stack_mgr: Stack port 2
on Switch 2 is up
000295: *Sep 30 21:52:21.455: %STACKMGR-5-SWITCH_ADDED: Switch 2 R0/0: stack_mgr: Switch 2 has been
added to the stack.
000296: *Sep 30 21:52:27.229: %IOSD_INFRA-6-IFS_DEVICE_OIR: Device usbflash1-2 added
000297: *Sep 30 21:52:28.217: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch 2:
EMP_RELAY: Channel UP!
000298: *Sep 30 21:54:24.621: %IOSXE_REDUNDANCY-6-PEER: Active detected switch 2 as standby.
000299: *Sep 30 21:54:24.620: %STACKMGR-6-STANDBY_ELECTED: Switch 1 R0/0: stack_mgr: Switch 2 has
been elected STANDBY.
000300: *Sep 30 21:54:29.635: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby insertion
(raw-event=PEER_FOUND(4))
000301: *Sep 30 21:54:29.635: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby insertion
(raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))
000302: *Sep 30 21:55:27.178: %HA_CONFIG_SYNC-6-BULK_CFGSYNC_SUCCEED: Bulk Sync succeeded
000303: *Sep 30 21:55:28.180: %RF-5-RF_TERMINAL_STATE: Terminal state reached for (SSO)Standby has
reached SSO hot
==========================================
Stage 2/2: Fast reloading Active(Switchover)
==========================================
Check fast reload support and verification on switch
000304: *Sep 30 21:55:37.479: %FED_IPC_MSG-5-FAST_RELOAD_COMPLETE: Switch 2 R0/0: fed: Fast reload
operation complete[1]: fast_rld_verify package(s) on switch 1
Finished preverifying before fast reload
SUCCESS to verify packages
SUCCESS to verify before fast reload
[1]: Finished fast_rld_verify successful on switch 1
   (-2) SUCCESS: Finished fast_rld_verify: Success on [1]
Extracting /tmp/vmlinux from /flash/cat9k-rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
6422576+0 records in
6422576+0 records out
6422576 bytes (6.4 MB, 6.1 MiB) copied, 15.8433 s, 405 kB/s
Extracting /tmp/initramfs.cpio.gz from /flash/cat9k-
rpboot.BLD_V171_THROTTLE_LATEST_20190906_004026_2.SSA.pkg
DBAL IOS: DBAL convert response to props failed: rc 102, error: 0
000305: *Sep 30 21:57:23.851: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch 2:
EMP_RELAY: Channel DOWN!
000306: *Sep 30 21:57:25.410: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch 2:
EMP_RELAY: Channel UP!38042177+0 records in
38042177+0 records out
38042177 bytes (38 MB, 36 MiB) copied, 94.1099 s, 404 kB/s
/usr/sbin/kexec -l /tmp/vmlinux --append="root=/dev/ram rw console=tty0,9600n8 max_loop=64
pciehp.pciehp_force unknown_nmi_panic BOARDID=20566 bdinfo_start=0xAC9CA018 bdinfo_size=0x3AC70
rd_start=0x8BCF7000 rd_size=0x2447DBF pkg_start=0x0 pkg_size=0x0  fastreload 1ru stack
SR_BOOT=flash:packages.conf  intel_pstate=disable" --ramdisk=/tmp/initramfs.cpio.gz
/bin/systemctl start kexec.target
Sep 30 21:57:35.307: %PMAN-5-EXITACTION: F0/0: pvp: Process manager is exiting:
Sep 30 21:57:36.667: %PMAN-5-EXITACTION: R0/0: vp: Process manager is exiting:
Sep 30 21:57:36.776: %INSTALL-5-INSTALL_COMPLETED_INFO: R0/0: install_engine: Completed install
one-shot PACKAGE flash:cat9k-eFSU-2.bin
Waiting for 120 seconds for other switches to boot
Switch number is 1
```

```
All switches in the stack have been discovered. Accelerating discovery
000174: *Sep 30 21:59:48.866: %IOSXE_REDUNDANCY-6-PEER: Active detected switch 1 as standby.
000175: *Sep 30 21:59:48.865: %STACKMGR-6-STANDBY_ELECTED: Switch 2 R0/0: stack_mgr: Switch 1 has
been elected STANDBY.
000176: *Sep 30 21:59:53.951: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby insertion
(raw-event=PEER_FOUND(4))
000177: *Sep 30 21:59:53.951: %REDUNDANCY-5-PEER_MONITOR_EVENT: Active detected a standby
insertion (raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))

C9300-FSU-Stack# show switch
Switch/Stack Mac Address : a0f8.4910.2400 - Local Mac Address
Mac persistency wait time: Indefinite
                                             H/W   Current
Switch#   Role    Mac Address     Priority Version  State
------------------------------------------------------------------------------------
---
 1       Standby  a0f8.4910.2400     14      V01     HA sync in progress
*2       Active   d4ad.bd9d.8600     1       V02     Ready


C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: On hold
Minimum required system uptime before fast reload can be supported is 900 seconds
Client OSPFV3                          : (0x10203007) Status: Up
Client OSPF                            : (0x10203006) Status: Up
Client GR_CLIENT_BGP                   : (0x10203005) Status: Up
Client IS-IS                           : (0x10203004) Status: Up
Client GR_CLIENT_TOPO                  : (0x10203003) Status: Up
Client GR_CLIENT_VRF                   : (0x10203002) Status: Up
Client GR_CLIENT_RIB                   : (0x10203001) Status: Up
Client GR_CLIENT_FIB                   : (0x10203000) Status: Up
000178: *Sep 30 22:00:48.801: %HA_CONFIG_SYNC-6-BULK_CFGSYNC_SUCCEED: Bulk Sync succeeded
000179: *Sep 30 22:00:49.803: %RF-5-RF_TERMINAL_STATE: Terminal state reached for (SSO)
000180: *Sep 30 22:00:57.993:  Unlocking Config. Fast Reload done
000181: *Sep 30 22:00:57.988: %STACKMGR-5-FAST_RELOAD_DONE: Switch 2 R0/0: stack_mgr: Fast
reload operation completed successfully on entire stack
000182: *Sep 30 22:00:58.986: %FED_IPC_MSG-5-FAST_RELOAD_COMPLETE: Switch 1 R0/0: fed: Fast
reload operation complete
```

Once the stack comes back online with prompt; please check the output of the below command. The protocol status should be in "*Up*" status as below. *It can generally take about 4-5 minutes.*

```
C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: Dataplane update granted
Minimum required system uptime before fast reload can be supported is 5 seconds
Client OSPFV3                          : (0x10203005) Status: Dataplane update granted
Client OSPF                            : (0x10203004) Status: Dataplane update granted
Client GR_CLIENT_BGP                   : (0x10203003) Status: Dataplane update granted
Client IS-IS                           : (0x10203002) Status: Dataplane update granted
Client GR_CLIENT_RIB                   : (0x10203001) Status: Dataplane update granted
Client GR_CLIENT_FIB                   : (0x10203000) Status: Dataplane update granted

C9300-FSU-Stack# show graceful-reload
Graceful Reload Infra Status: Not running
Minimum required system uptime before fast reload can be supported is 5 seconds
Client OSPFV3                          : (0x10203005) Status: Up
Client OSPF                            : (0x10203004) Status: Up
Client GR_CLIENT_BGP                   : (0x10203003) Status: Up
Client IS-IS                           : (0x10203002) Status: Up
Client GR_CLIENT_RIB                   : (0x10203001) Status: Up
Client GR_CLIENT_FIB                   : (0x10203000) Status: Up
```

Stop the Pings on the Data Switch by hitting CTRL-Shift -6 and observe/calculate the traffic loss. Total traffic downtime should be less than 30 sec.

Task 4: Clearing the file system

Step 1: Clear the files in the flash.

Since xFSU is only working in '*Install mode*' and there will be a lot .pkg file after software update. You may want to clear the files from previous version of software code.

```
C9300-FSU-Stack# install remove inactive
install_remove: START Fri Oct 12 00:45:25 UTC 2019
Cleaning up unnecessary package files
No path specified, will use booted path flash:packages.conf
Cleaning flash:
  Scanning boot directory for packages ... done.
  Preparing packages list to delete ...
    cat9k-cc_srdriver.BLD_V1610_THROTTLE_LATEST_20181003_010707_2.SSA.pkg
      File is in use, will not delete.
……..
      File is in use, will not delete.
    cat9k-wlc.BLD_V1610_THROTTLE_LATEST_20181003_010707_2.SSA.pkg
      File is in use, will not delete.
    packages.conf
      File is in use, will not delete.
  done.
The following files will be deleted:
[switch 1]:
/flash/cat9k-cc_srdriver.BLD_V1610_THROTTLE_LATEST_20180927_010858.SSA.pkg
/flash/cat9k-espbase.BLD_V1610_THROTTLE_LATEST_20180927_010858.SSA.pkg
/flash/cat9k-guestshell.BLD_V1610_THROTTLE_LATEST_20180927_010858.SSA.pkg
…..
/flash/cat9k_iosxe.FSU_1.1.conf
/flash/cat9k_iosxe.FSU_1.2.conf
/flash/cat9k_iosxe.FSU_1.bin
….
/flash/cat9k_iosxe.FSU_2.bin
/flash/cat9k_iosxe.FSU_2.conf
Do you want to remove the above files? [y/n]y
```

Summary

Extended Fast software upgrade feature significantly reduces the traffic downtime during a software upgrade on both 9300 standalone and Stack.