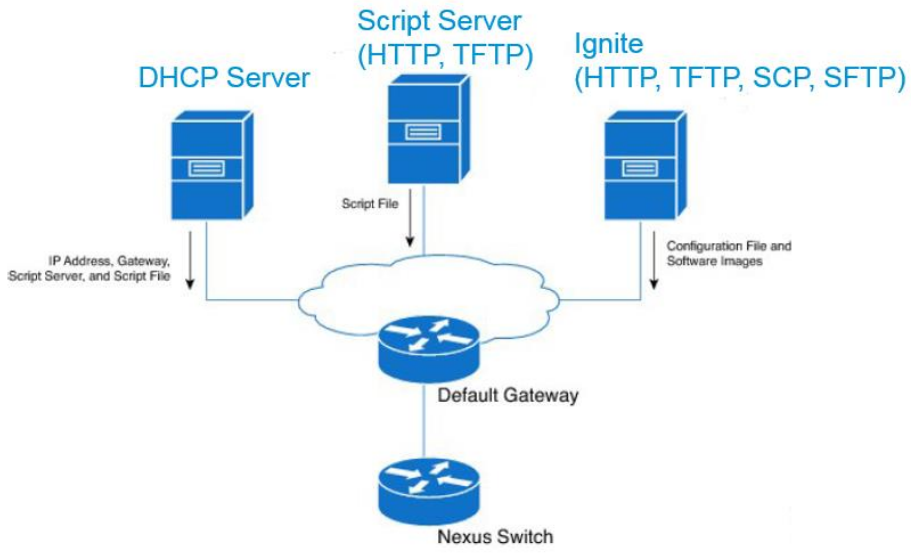
Ignite

Ignite automates the process of installing and upgrading software images and installing configuration files on Cisco Nexus 9000 and 3000 Series switches that are being deployed in the network.

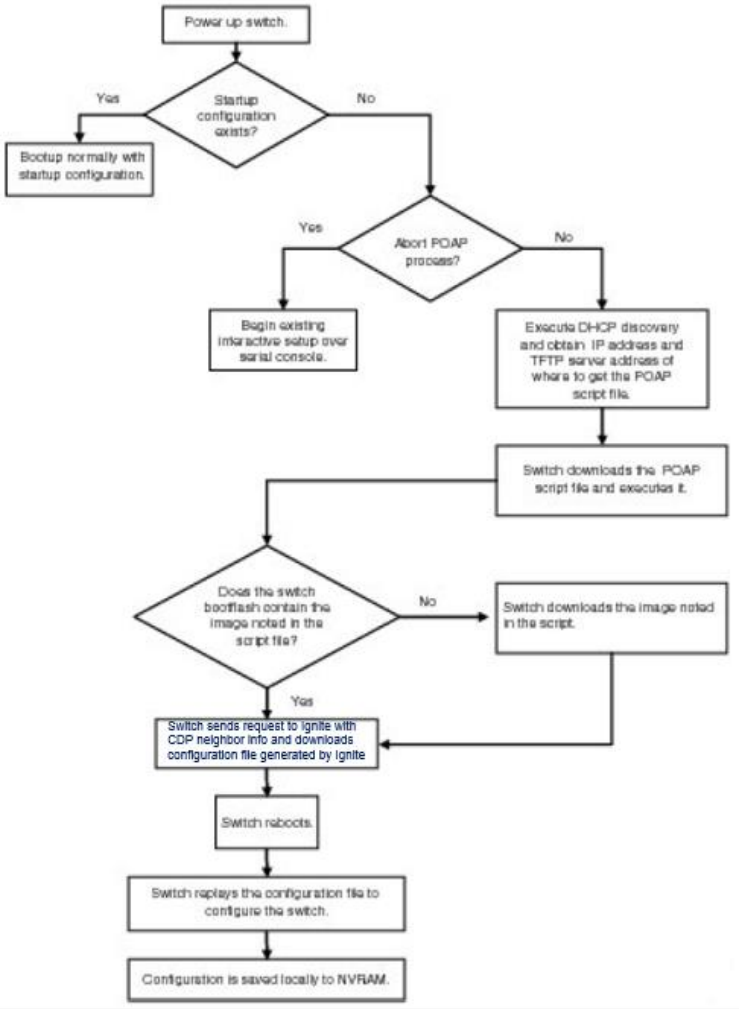
Overview

Cisco Nexus Power On Auto Provisioning using Ignite requires the following Network Infrastructure.

- A DHCP server to bootstrap the interface IP address, gateway address, and Domain Name System (DNS) server.
- A TFTP server that contains the configuration script used to automate the software image installation and configuration process. It is recommended that this script server also be installed in server running Ignite. Ignite_poap.py – startup script needed for interaction with Ignite is available in github repository.
- A server running Ignite that contains the desired software images and rules to dynamically build configuration files.



The Cisco Nexus POAP process has the following phases:



1. Power up

   When you power up the device for the first time, it loads the software image that is installed at manufacturing and automatically enters a DHCP-based POAP discovery phase.

2. DHCP discovery

   The switch sends out DHCP discover messages on the front-panel interfaces or the MGMT interface that solicits DHCP offers from the DHCP server or servers. The DHCP client on the Cisco Nexus switch uses the switch serial number/MAC address in the client-identifier option to identify itself to the DHCP server. DHCP server assigns an IP address and responds with a DHCP offer, which includes other options to support the switch bootstrapping (example: script server, script filename to get the bootstrap script). Please refer to Cisco POAP documentation *http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/6-x/fundamentals/configuration/guide/b_Cisco_Nexus_9000_Series_NX-OS_Fundamentals_Configuration_Guide/b_Cisco_Nexus_9000_Series_NX-OS_Fundamentals_Configuration_Guide_chapter_011.html* for additional details:

   > Script server name or Script server address—The DHCP server relays the Script server name or Script server address to the DHCP client. The DHCP client uses this information to contact the Script server to obtain the script file. TFTP or HTTP may be used to download the script file.
   >
   > Script file name—The DHCP server relays the script file name to the DHCP client. The boot file name includes the complete path to the script file on the Script server. The DHCP client (switch) uses this information to download the script file. To take advantage of the dynamic configuration generation capabilities of Ignite server a special bootstrap script named “ignite_poap.py” is available.

3. Script execution

   After the device bootstraps itself using the information in the DHCP acknowledgement, the script file is downloaded from the Script server.

   This script (*ignite_poap.py*) should be loaded in Script server for the switch to download it in this phase (*Note: best practice is to run the Script server on the Ignite VM*).

When switch executes this script, it enables the switch interfaces and collects CDP neighbor adjacencies. The CDP neighbor information along with the system serial number, MAC address is then sent to Ignite in an HTTP GET request.

Ignite, acting as a server that handles these requests, attempts to identify this specific switch in its database, using the information provided in the request. If successful, the server builds configuration file for this switch and provides file details in its response. Details of how Ignite identifies the switch and builds the configuration file are explained in later sections.

During script execution, all events and errors are logged by the script both locally in the bootflash as well as in Ignite.

The ignite_poap.py script also downloads and installs the software image appropriate for this switch.

The configuration file retrieved from Ignite saved as the “scheduled configuration” to be applied when the switch boots up with the correct image. On reload, the switch boots up with the new software image and the scheduled config is applied.

4. Post-installation reload

   On successful application of the scheduled config as the running config, the switch copies the running configuration to the startup configuration.

Ignite

Ignite provides a powerful and flexible framework through which a data center network administrator can define datacenter network in terms of topology, data center fabric, repository of switch discovery rules, configuration templates (configlets), configuration scripts and resource pools used to allocate IP Addresses/VLANs/identifiers used in provisioning network features. Ignite server automatically identifies the switch from its neighbors and links connecting to its neighbors, matching it with fabric topology

information provided by the administrator. Ignite server then builds a configuration unique to this switch from rules provided by administrator using configlets, scripts and resource pools filling in with values provided in the fabric.

Environment Setup Required for Ignite

DHCP, TFTP services can also be setup on Ignite server in addition to running the Ignite POAP service. It is also possible to run these services on other existing servers. Following setup/configuration are needed in the DHCP and TFTP servers, for the POAP process through Ignite server to function:

Install Ignite Server

Follow instructions from github repository https://github.com/datacenter/ignite to install the Ignite application. After installing the application, verify that you are able to access Ignite User Interface.

DHCP setup/configuration

1. Lease time is set to a high value (3600 minutes)
2. Option tftp-server-name is set to IP address of the TFTP server where initial boot script file will reside.
3. Option bootfile-name is set to “ignite_poap.py”. This file is generated by Ignite server and will be available in the directory where you installed ignite server application. This file should be copied to TFTP server (referred above) and bootfile-name should have the path relative to working directory of TFTP service. For example, if tftpd is started from directory /var/lib/tftpboot and if “ignite_poap.py” is stored in /var/lib/tftpboot director, bootfile-name is set to “ignite_poap.py”.

Note: if you are setting up a windows server as a TFTP server, option 067 Bootfile Name should be set to “ignite_poap.py” with path details and option 150 TFTP Server should have the TFTP server IP address.

Note: please always use "ignite_poap.py" created during Ignite POAP server installation. This file is set up with all Ignite server information required by the switch to get startup configuration and image.

Image Server

Prepare the image server for use with Ignite POAP. This server should have all the required switch, kickstart and EPLD images stored. Depending on the transport protocol in use, the server should be running the appropriate service "tftp" or "ftp" or "scp" or "http". All required images should be downloaded to this server and ready for use.

It is recommended that you use Ignite POAP server as your image server as well.

Setup Switches and Images

Switch models and switch images used in the data center environment served by Ignite server is required to be defined, prior to switches can be booted up through Ignite POAP. Pre-defined switch details are created and available during Ignite server install. These details can be modified or new switch models can be defined to suit the environment.

Image profiles define images downloaded and available for use. Image profile contains image path and file name, server address where image resides, transfer protocol to be used for downloading image file and credentials required for downloading. Image profiles will be used during POAP process. In response to a switch requesting startup configuration, image profile to be used to boot the switch will also be provided. Details of how to associate image profiles with a switch are discussed later.

Switch model is defined using following properties:

Property	Sub-Property	Description
Name		User defined name which will be used to identify the switch
Base Model		Cisco Model number which identifies the base switch being defined. Nexus 9000 switches come in fixed and chassis configurations. Some fixed switches support plugin modules. Chassis switches support multiple line cards each of which could be plugged into a slot.
Tier Role		Role assigned to this model in a fabric topology. Fabric tier roles are Spine, leaf, border router, core router. If this model can be used in different roles, select multiple roles.
Type		Fixed or Chassis
Fixed Switch Properties		
Port Group • Group of ports with similar properties	Number of ports	Number of ports of a category
	Port Speed	Speed of the ports 1/10G, 40G , 100G
	Transceiver	GBASE-T, SFP+, QSFP+, QSFP2B, CFP2

• Multiple port groups can be defined • Port numbers are automatically generated by Ignite based on slot/module, speed etc. •	Port Roles	Role in which this group of ports are used – Uplink, Downlink, Both
Module (Optional)		Plugin module present in the fixed switch. This module should also be defined in the switch database before it can be used.
Chassis Switch Properties		
Number of slots		Total number of slots available for line cards
Slot details (Slot details should be filled for each slot where a line card is present)	Slot No.	Slot number where the card resides
	Line Card	Line Card present in the slot. This line card should also be defined in the switch database before it can be used.

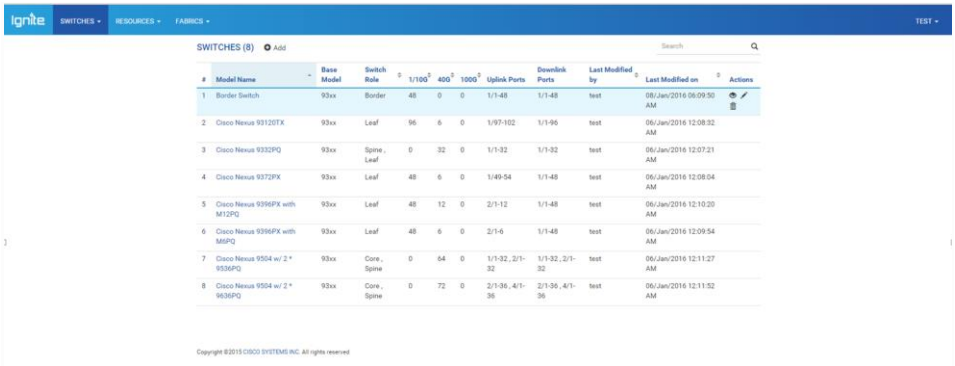
Line Card / Modules used in switches are defined using following properties:

Property	**Sub-Property**	**Description**
Model Name		User defined name which will be used to identify the switch
Type		Line Card or Module
Module / Line Card ports		
Port Group	Number of ports	Number of ports of a category
	Port Speed	Speed of the ports 1/10G, 40G , 100G

• Group of ports with similar properties • Multiple port groups can be defined • Port numbers are automatically generated by Ignite based on slot/module, speed etc.	Transceiver	GBASE-T, SFP+, QSFP+, QSFP2B, CFP2
	Port Roles	Role in which this group of ports are used – Uplink, Downlink, Both

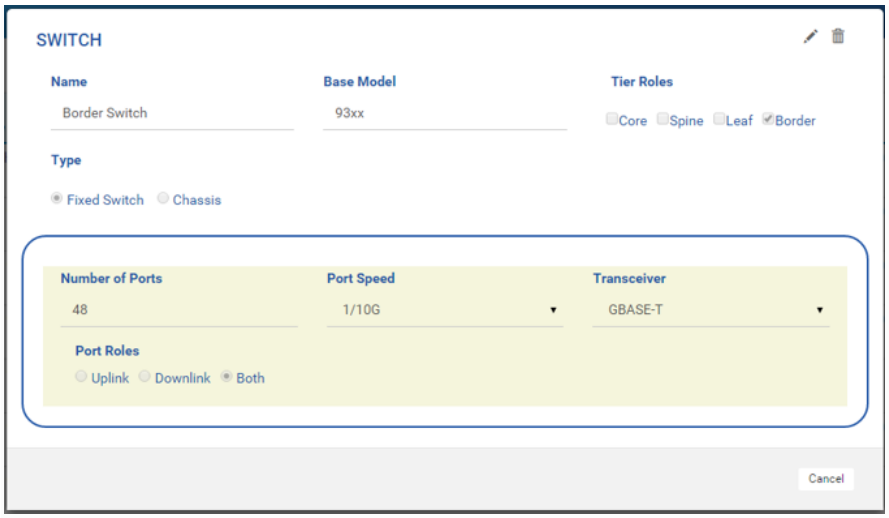
Viewing, Adding and Editing Switches

1. In “Switches” tab from top menu, select option “Switches”
2. List of switches is displayed

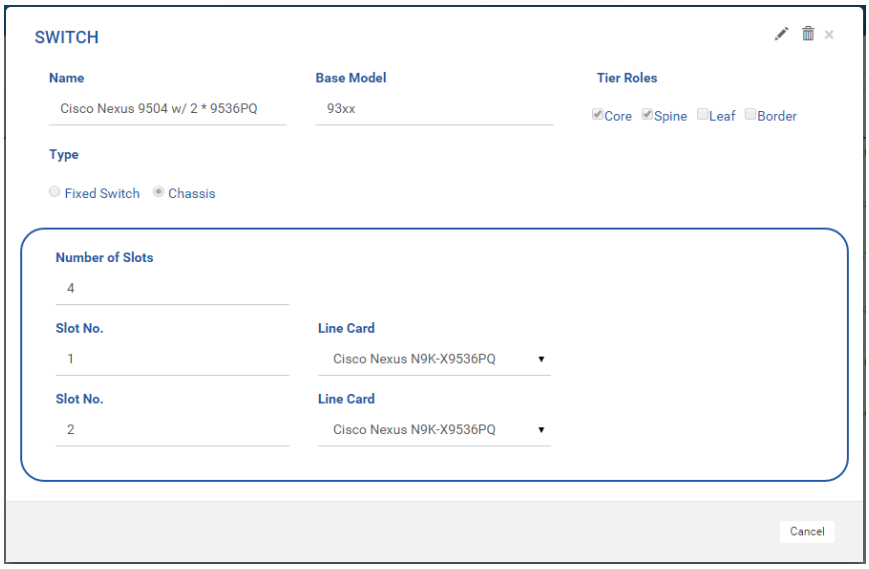


3. To view switch details, select a switch from list and click.

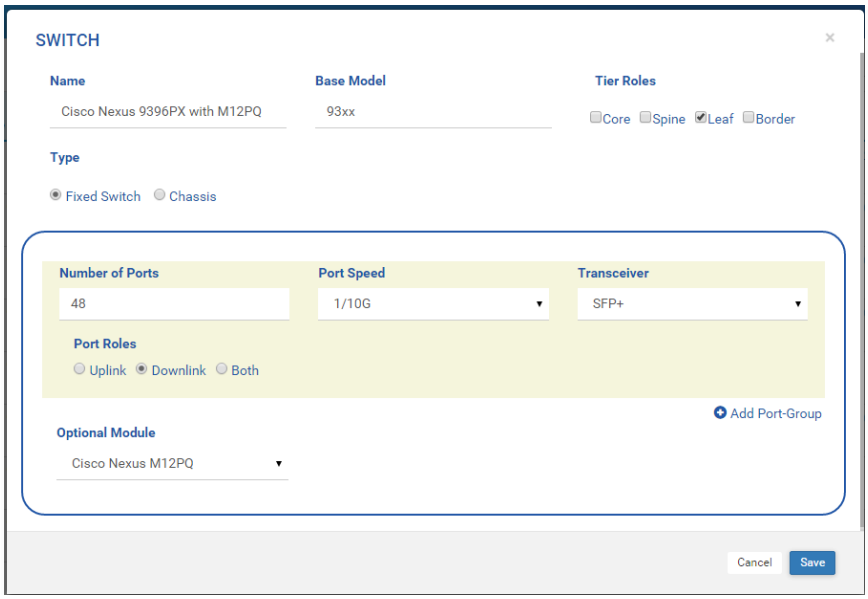
Fixed switch



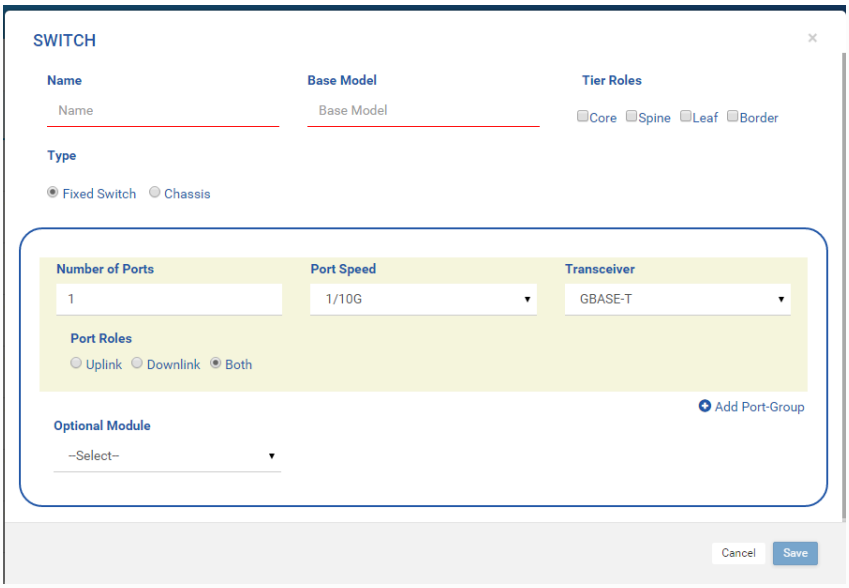
Chassis Switch



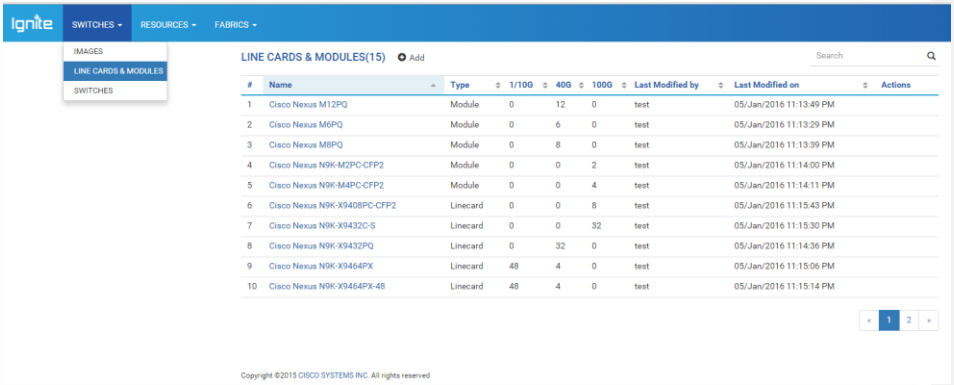
4. To edit switch details, click on edit icon, modify. Click save to save changes.
   Note: changes are not accepted if switch is used in a topology / fabric.



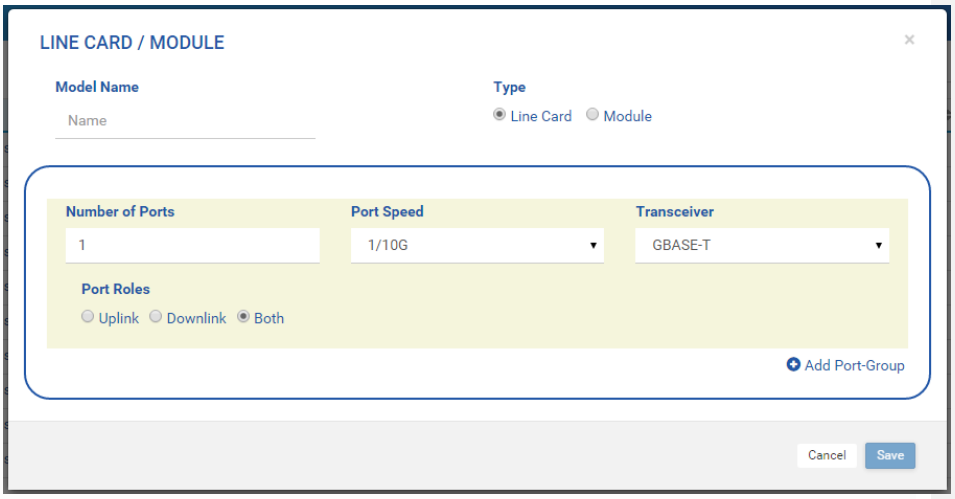
5. To add a new switch, click +Add button, in list view, enter switch details and save.



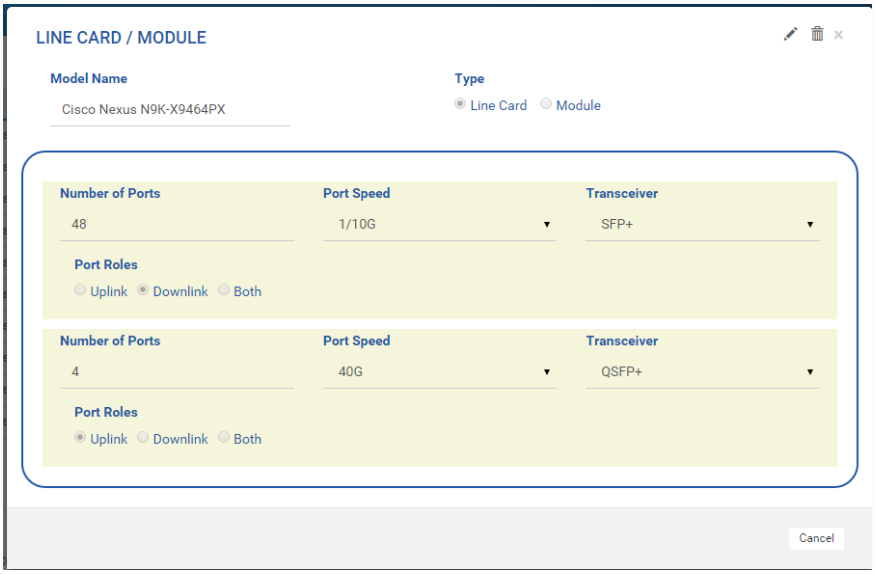
6. To view, edit or add line cards / modules select “LINE CARDS & MODULES” option in the SWITCHES tab. A list of line cards and modules is displayed.



7. New line cards can be added by clicking “+ Add” button and filling details in the line card/ module form.



8. To view details of a line card or module, select a card from list view and click.



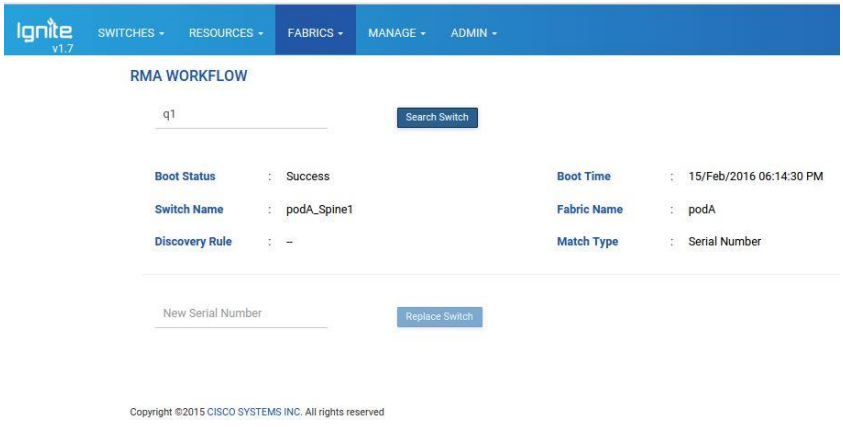
RMA (Return Merchandise Authorization):

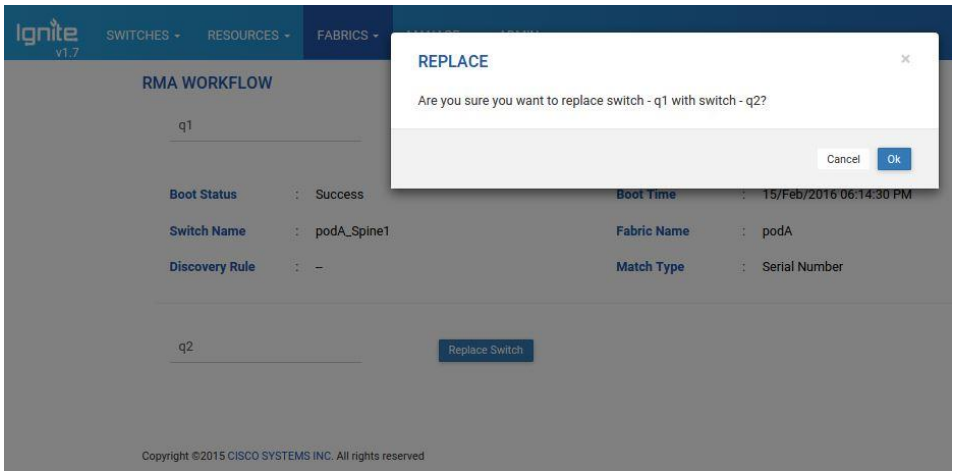
RMA is defined as a process for replacing the serial number of a switch whenever any switch is being replaced by a new switch.

RMA consists of two parts:

In the first part, old serial number is being searched. In response to this search, complete information about that serial number is returned to the user i.e. fabric information or discovery rule information.

In the second part, the old serial number can be replaced with new serial number. It will result in replacement at all respective places in database.





During the search operation of old serial number, there can be following possibilities:

- Match is found in fabric(fabric_switch) only and the switch booting status is SUCCESS or FAILURE
- Match is found in fabric(fabric_switch), the switch is not booted but BUIL CONFIG is done
- Match is found in fabric(fabric_switch), the switch is not booted and BUILD CONFIG is not done
- Match is found in fabric(fabric_switch), switch is booted by matching into discovery rules with match type as ‘serial number’
- Match is found in fabric(fabric_switch), switch is booted by matching into discovery rules with match type as ‘Neighbor’
- Match is found in only Discovery Rule (the switch is not booted)
- Match is found in fabric(fabric_switch) and Discovery rule also exist - when RMA is performed more than one time
- Match is found in fabric(fabric_switch) by matching into fabric/discovery rule but the booting of the switch is in Progress (In this case an error is returned)
- Match is not found anywhere (Error is returned)

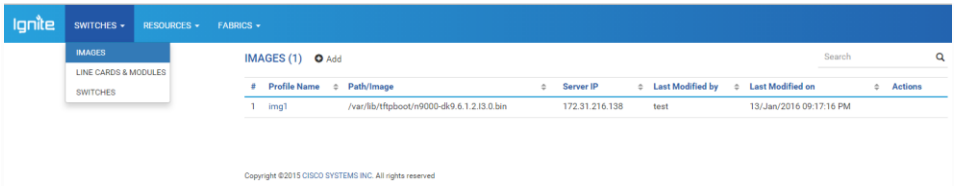
If old serial number matches with any of the non-error cases listed above, replacement is performed at all respective places i.e. fabric, discovery rule, etc.

Setup Image Profile

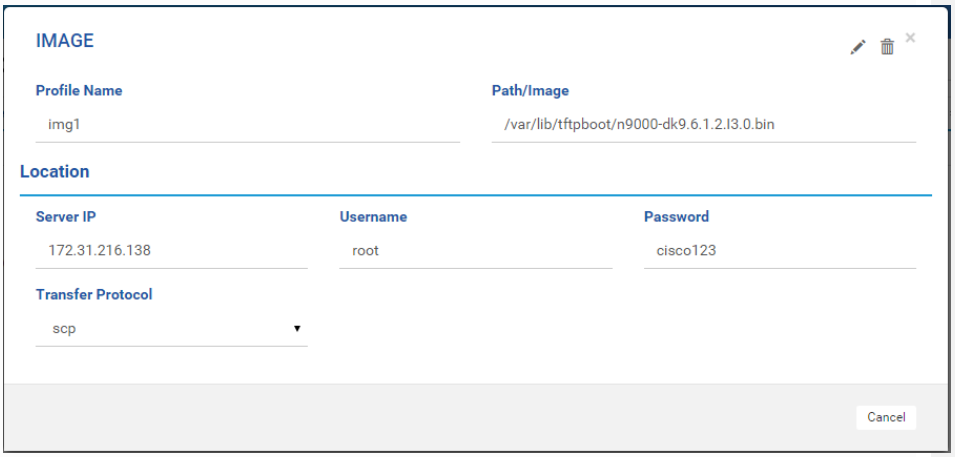
Image profile defines image name with directory path, server in which image is available, transport protocol to be used to download image file, user credentials required for download.

To edit, view or add image profiles select “IMAGES” option from “SWITCHES” tab in top menu. List of image profiles is displayed.

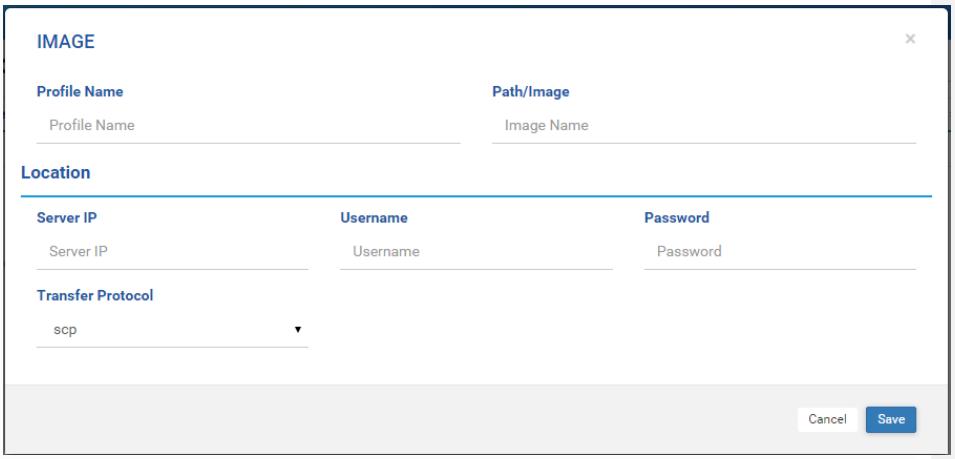
1. View Image profiles – select “SWITCHES” and “IMAGES” option.



2. Select an image from list to view or edit.



3. To add a profile, click “+Add” button in list view and enter details in IMAGE form.



Setup Configuration Profile

Configuration Profile is used to build a startup configuration for a switch during POAP process.

Configuration profiles use CONFIGLETs to assemble a startup configuration. CONFIGLETs can be of 2 types: (a) CLI template (b) Python script. A configlet when processed by Ignite server during the configuration profile assembly/build, will generate CLI commands to be stored in startup configuration.

CONFIGLETS

Configlets can be parameterized. Parameterization allows configlet be generic so that it can be used in multiple configuration profiles. In each configuration profile configlet parameter can be assigned different values, so that it generates CLI configured specific to a switch.

A simple example to illustrate this capability would be a CONFIGLET required to assign management IP address. Example CONFIGLET to generate this configuration:

```
interface mgmt0
vrf member management
no cdp enable
ip address $$MGMTIPADDRESS$$/22
no shutdown
```

Notice that in CLI, “ip address $$MGMTIPADDRESS$$/22”, actual value of IP address is not provided. Ignite recognizes, “$$MGMTIPADDRESS$$” as a parameter required for this configlet and whenever this configlet gets used in a CONFIG PROFILE, user will be asked to enter a value.

CONFIGLET can also be a Python script. For example, when same CLIs have to be applied to a set of interfaces, it is better to have a script generate these CLIs, instead of repeatedly entering these commands for every interface.

```
for INTERFACE in [“ethernet1/1”, “ethernet1/2”, “ethernet1/3”]
    print “interface %s” %INTERFACE
    print “no switchport”
    print “ip pim sparse-mode”
    print “ip router ospf 1 area 0.0.0.0”
    print “no shutdown”
```

CONFIGLET script above will generate configurations for 3 interfaces named “ethernet1/1”, “ethernet1/2”, and “ethernet1/3”.

In the example above where a CONFIGLET uses a parameterized value and requires that value to be assigned during configuration build process, actual value could be assigned from a pool. User can define pools and use these pools to assign parameter values dynamically.

POOL

Pool is a collection of values which are managed by Ignite server for allocation on request to replace parameters with actual values. For example, in the CONFIGLET which assigns a management IP address to switches, parameter value $$MGMTIPADDRESS” could be dynamically assigned from a POOL.

Ignite keeps track of the assignment of values from the pool. As a value gets assigned Ignite updates the POOL with identity of the switch to which this value is associated with and the time at which assignment was made.

POOL can be created with IP address values or Integer values. You can also define scope of the POOL when you define it. Scope could be “Global” or “Fabric”. “Global” scope creates one pool and values assigned from this same pool for all requests. “Fabric” scope creates multiple pools of the same type per Fabric. This allows the reuse of values in different data center pods. For example, you have a VLAN pool and you want to maintain uniqueness of VLAN within one data center pod, but want to reuse the same VLAN values in another data center pod. You can define one POOL item for VLAN and assign scope as “Fabric”. Ignite will create multiple pools with the same pool definition for every “FABRIC” defined in the system.

CONFIG PROFILE

CONFIG PROFILE is used to define the assembly/build process of a startup configuration. CONFIG PROFILE is defined using a sequence of CONFIGLETs. When a CONFIGLET is added to a CONFIG PROFILE, user should define a method to associate a value with all the parameters used in that CONFIGLET. One method as explained is to use a pool value. The other methods available are:

Type	Description
FIXED	constant value e.g. FIXED.101 (value 101 will be assigned to the parameter)
POOL	a value allocated by the server from a pool of values defined by administrator e.g. POOL.MGMT_IP (a value from the POOL named MGMT_IP will be assigned to the parameter)
INSTANCE	a value generated from fabric topology (switch names, port numbers and IP addresses) e.g. INSTANCE.HOSTNAME (a value from FABRIC definition) ***Note: FABRIC is explained later***
VALUE	value referenced by the parameter name (typically used when the parameter value has already been associated from an earlier assignment) e.g if MGMTIPADDRESS = POOL.MGMT_IP is already assigned to parameter MGMTIPADDRESS, to reuse the same

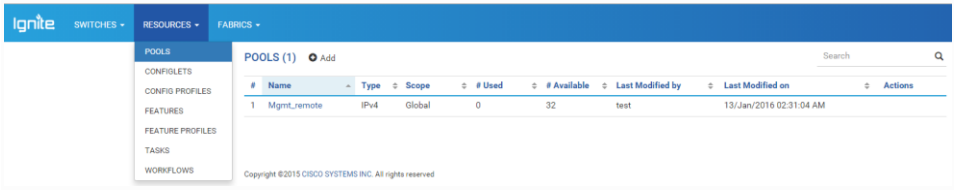
	value in another CONFIGLET, you can use VALUE.MGMTIPADDRESS

When CONFIGL PROFILE is processed during configuration build, CLIs will get generated in the same sequence in which CONFIGLETs are added to CONFIG PROFILE.

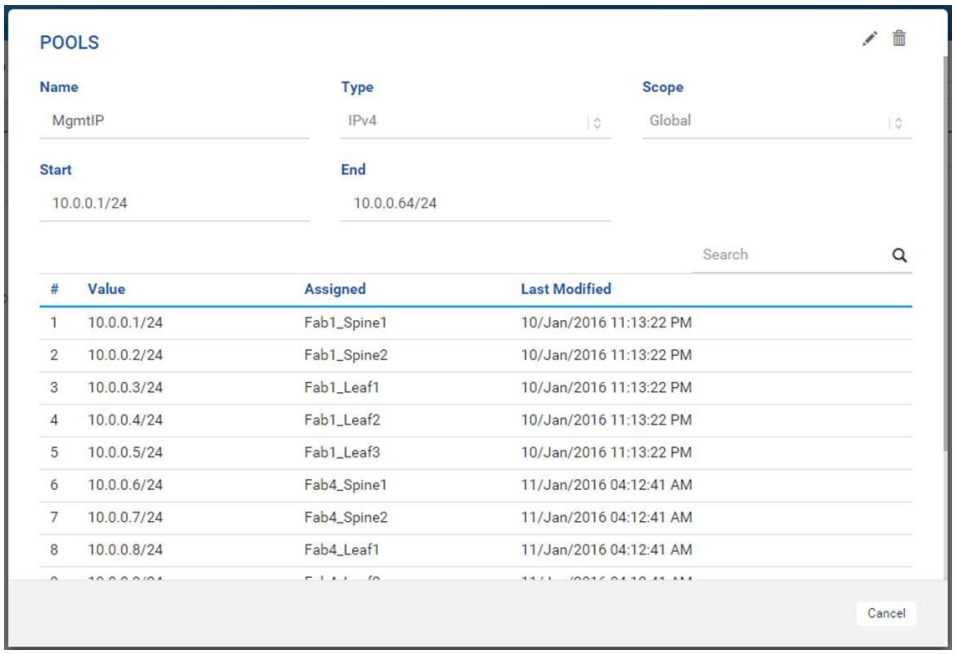
View, Edit, Add POOL

POOL operations are done by selecting the option “POOLS” from “RESOURCES” tab in the top menu.

A list of POOLs already defined in system is displayed.

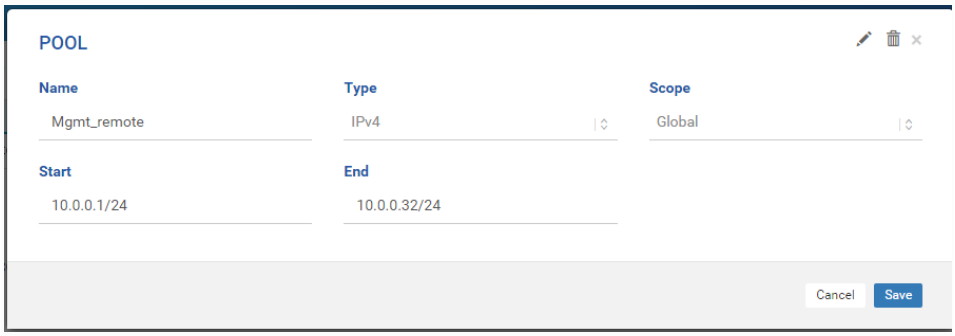


To view details of a specific POOL item, click on the name from the list.



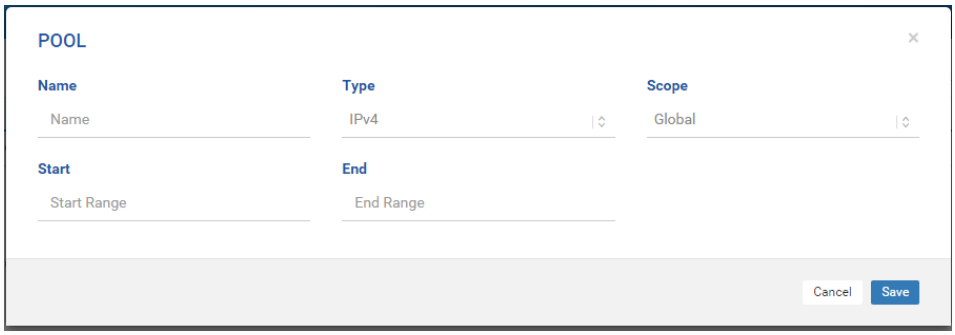
Pool values are displayed along with entries already in use by different switches.

To edit a POOL item, select edit icon from POOL item display, make changes and save.



To add a POOL item, click “+Add” button in list view, fill values and save.

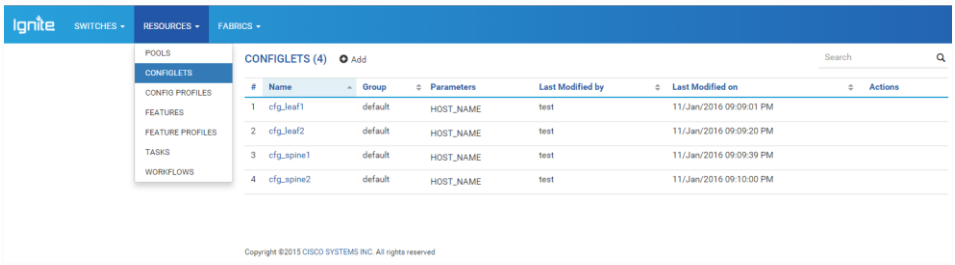
Type can be “IPv4”, “IPv6” or “Integer”. Scope can be “Global” or “Fabric”.



View, Edit or Add CONFIGLETS

CONFIGLETS operations are done by selecting the option “CONFIGLETS” from “RESOURCES” tab in the top menu.

A list of CONFIGLETs already defined in system is displayed.



To view details of a CONFIGLET, select the CONFIGLET from list by clicking it.

Code associated with CONFIGLET and the parameters extracted from the code are displayed. Parameters are identified by strings enclosed in delimiter “$$”. For example, a string $$HOST_NAME$$, will be identified as a parameter by Ignite server when the CONFIGLET code is uploaded.

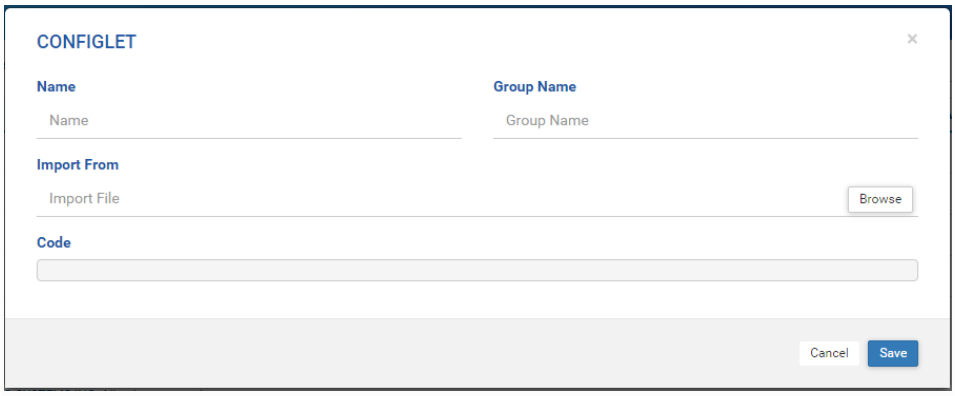
To edit a CONFIGLET, click on the edit icon, make changes and save.

Note: CONFIGLETs are text files imported from user systems. Any change required in the code should be applied to the file from which this CONFIGLET was imported. After making these changes, edit CONFIGLET and import the update.

Note: if a CONFIGLET is in use, in a CONFIG PROFILE, changes done to CONFIGLET will take effect only when the next request for boot is received from a switch to which this CONFIG PROFILE is applied.

To add a CONFIGLET, click “+Add” button from CONFIGLET list view.

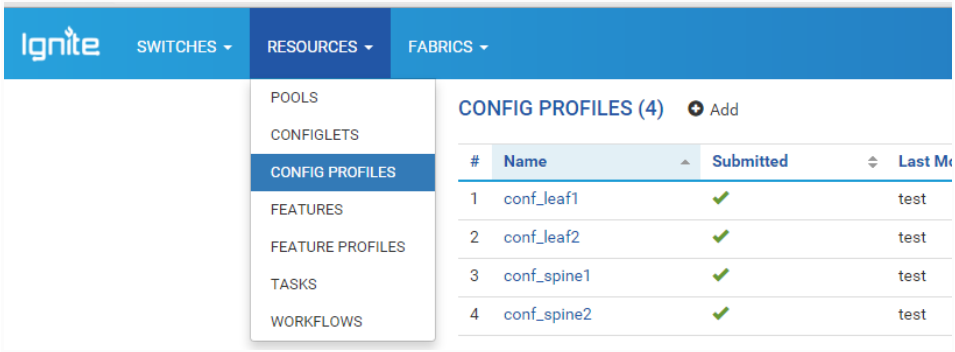
Enter values and save. To classify CONFIGLETs according to its functionality, a GROUP NAME can be associated with a CONFIGLET. For example, a GROUP NAME “Layer2” could be used to categorize CONFIGLETs which generate CLIs related to Layer 2 functionality. GROUP NAME is a free form text string, no meaning is associated with this text.



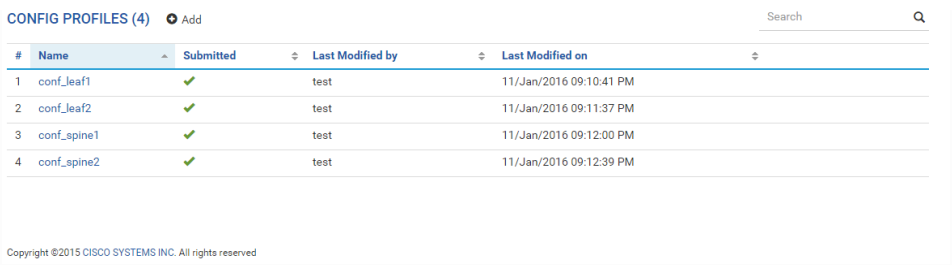
CONFIGLET code is imported from text files stored in user system (system from where the UI is running). Ignite parses the code and identifies parameter strings once the code is imported and saved.

View, Edit or Add CONFIG PROFILE

CONFIG PROFILE operations are done by selecting the option “CONFIG PROFILES” from “RESOURCES” tab in the top menu.

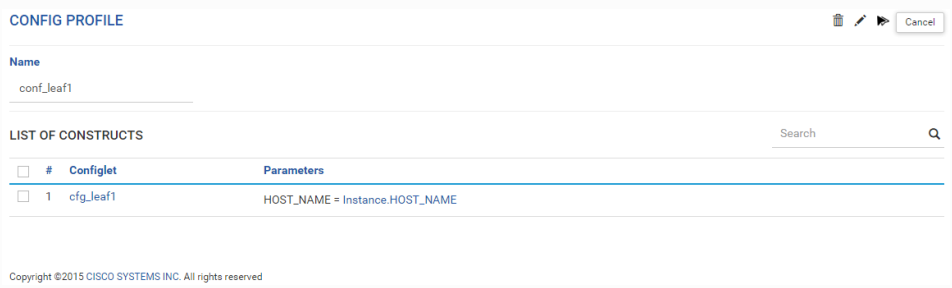


A list of CONFIG PROFILEs already defined in system is displayed.

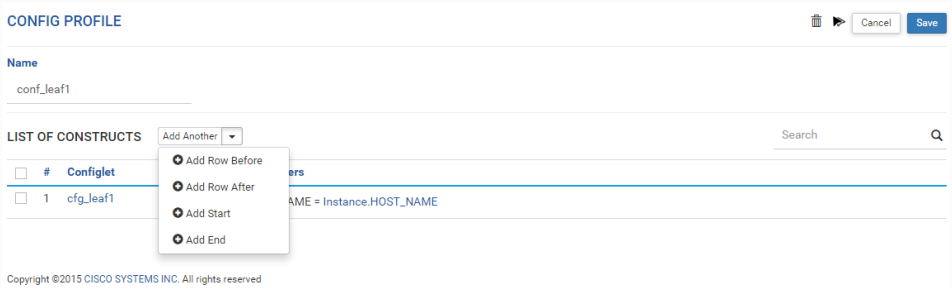


To view details of a CONFIG PROFILE, select the CONFIG PROFILE from list by clicking it.

CONFIGLETs are listed in the order in which they were added to CONFIG PROFILE along with its parameters and values assigned.



To edit a CONFIGLET, click on the edit icon, make changes and save.



You can add, delete CONFIGLETs in CONFIG PROFILE during edit process.

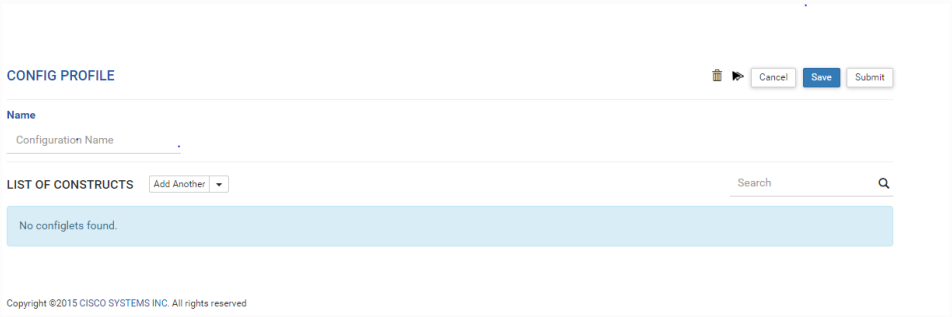
Following operations are allowed:

Add Row Before – allows adding a CONFIGLET before a selected row in the CONFIGLET list
Add Row After – allows adding a CONFIGLET after a selected row in the CONFIGLET list
Add Start – allows adding a CONFIGLET at the start of CONFIGLET list
Add End – allows adding a CONFIGLET at the end of CONFIG list

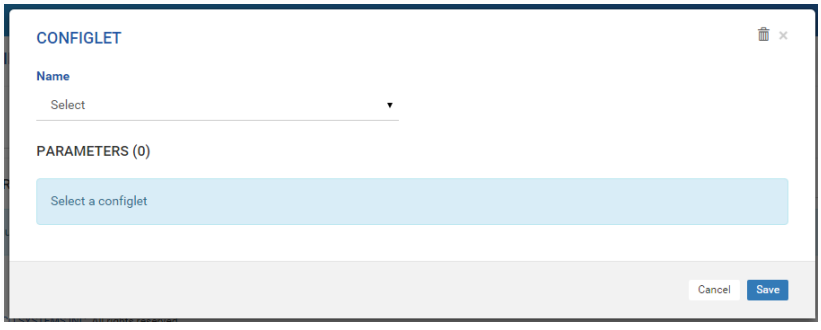
A row is selected for operation, by clicking on the selection box.

Note: if a CONFIG PROFILE is in use, changes done to CONFIG PROFILE will take effect only when the next request for boot is received from a switch to which this CONFIG PROFILE is applied.

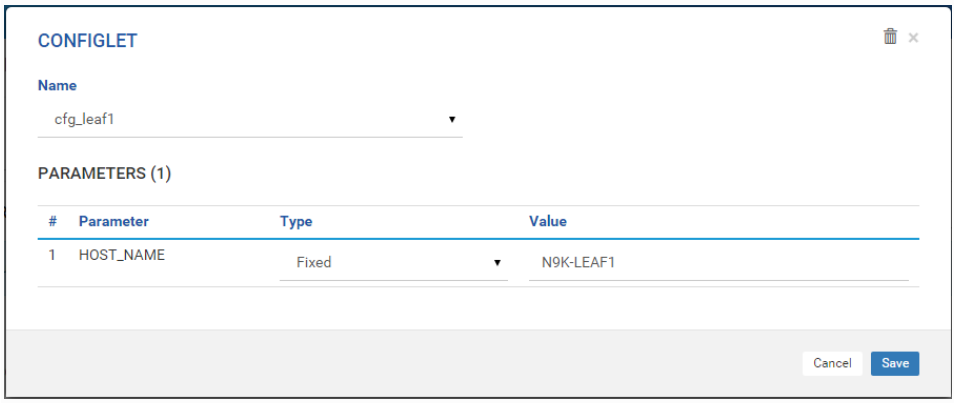
To add a new CONFIGLET, click “+Add” button from CONFIGLET list view.



Assign Name to CONFIG PROFILE, define the list of CONFIGLETs used to build this configuration by selecting “Add Another” button and selecting a CONFIGLET from list of available CONFIGLETs.

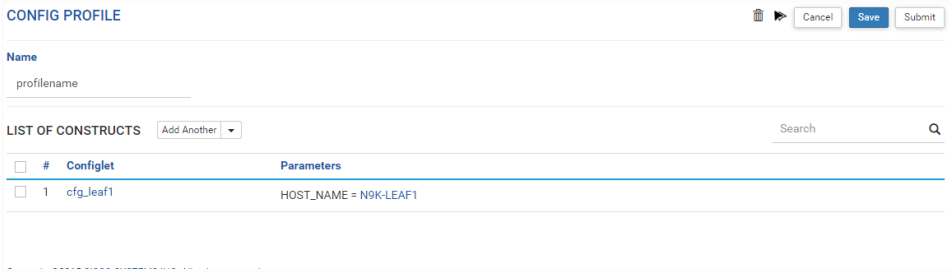


Once a CONFIGLET is added, all the parameters for that CONFIGLET will be listed and a value should be assigned to each parameter. Method of assigning parameter values is selected first and then the value per the selected method is assigned.



Once all parameters are assigned, click SAVE to add the selected CONFIGLET in the CONFIG PROFILE.

After adding all CONFIGLETs click “Submit” button on right top to submit this CONFIG PROFILE and make it available for use in switches.



Setup Discovery Rules

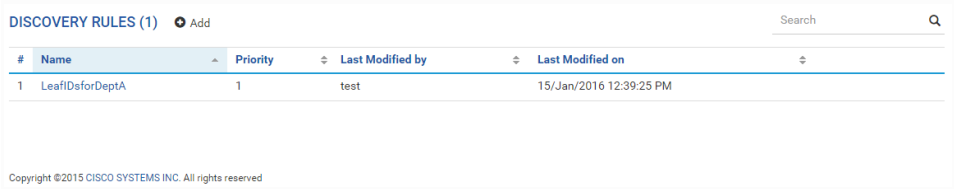
Discovery Rules are used to identify a switch and associate IMAGE PROFILE and CONFIG PROFILE to it. Once an IMAGE PROFILE and CONFIG PROFILE is associated, Ignite server is ready to process the POAP request and send the appropriate image and startup configuration to the switch.

A switch is identified in Ignite by its Serial ID or through neighbor information. Discovery Rules allow you to set up this information in Ignite.

View, Edit, Add Discovery Rules

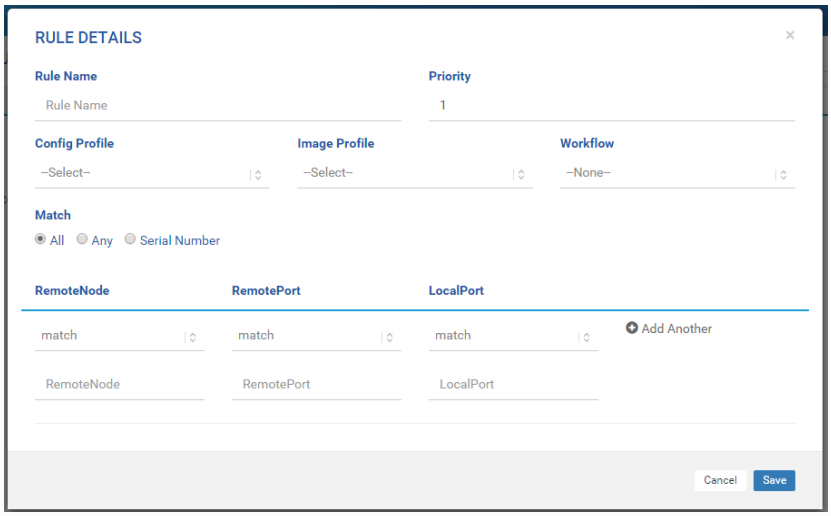
Discovery rule operations are done by selecting “DISCOVERY RULE” option in “FBARICS” tab.

List of DISCOVERY RULEs defined is displayed.



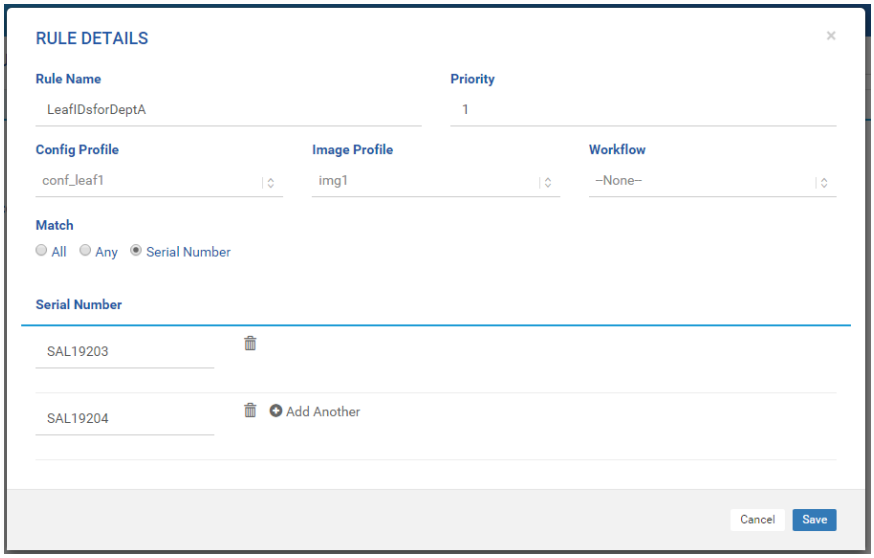
To add a new DISCOVERY RULE, click “+Add” to add a new discovery rule.

Edit or delete operations can be performed by selecting the DISCOVERY RULE from list and clicking edit or delete icons.



Creating DISCOVERY RULE with Switch Serial Number(s)

To create a new DISCOVERY RULE and associate it with switch Serial Number(s) can be done by selecting “Serial Number” option in RULE DETAILS page for “Match”.



Following fields in RULE DETAILS form should be filled and saved.

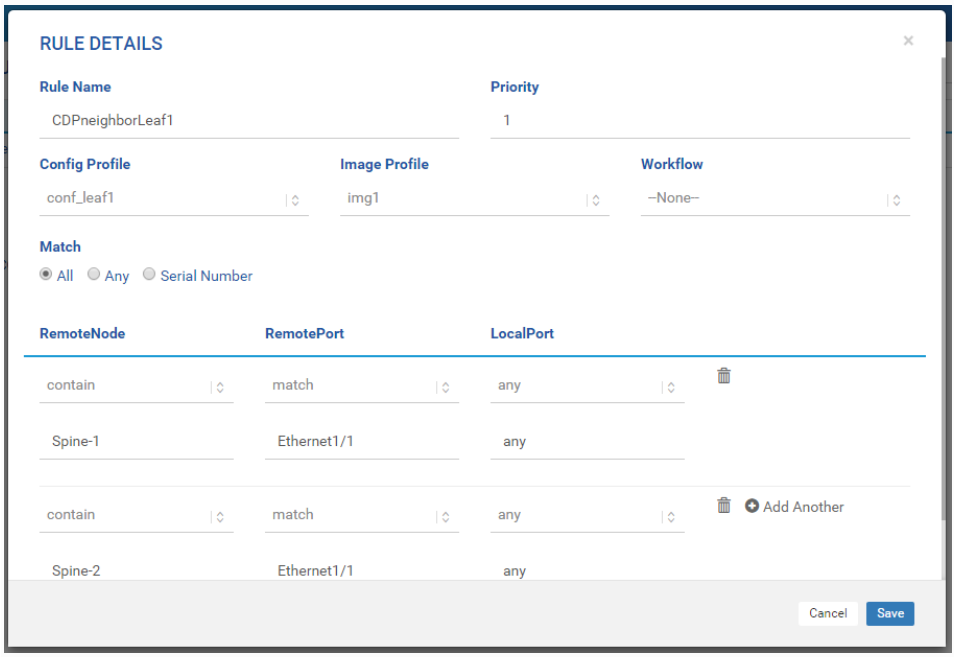
Filed	Description
Rule Name	Assign a unique name for this rule
Priority	A sequence number used in searching through discovery rules. Ignite server searches through discovery rules starting from the lowest priority value to highest priority value. First match will be used for further processing.
Config Profile	Name of CONFIG PROFILE used to build the startup configuration. Values can be filled from the drop down available in this option. Only CONFIG PROFILEs submitted, will be available for selection.
Image Profile	Name of IMAGE PROFILE with switch image details to be sent to the switch to boot. Values can be filled from the drop down available in this option.
Workflow	Additional user defined tasks to be run during POAP boot process in the switch, ***Note: Workflows are explained in Advanced Configuration section.***
Match	Select "Serial Number" to define a discovery rule matching switch Serial Numbers.
Serial Number(s)	Serial Number of the switch to which this rule applies. Multiple serial numbers can be added in the rule. CONFIG PROFILE and IMAGE PROFILE from this discovery rule are applied to all serial numbers assigned in this DISCOVERY RULE.

Creating DISCOVERY RULE with neighbor information

CDP neighbor information is reported by the switch sending its POAP boot request to Ignite. This data can be used to identify the switch, instead of using a serial number.

Discovery Rule allows configuring multiple match expressions, to compare the remote node, remote port and local port in CDP data. A rule match is successful either with

“All” match expressions successful or “Any” match expression successful. “All” or “Any” can be selected during DISCOVERY RULE creation.



Following fields in RULE DETAILs form should be filled and saved.

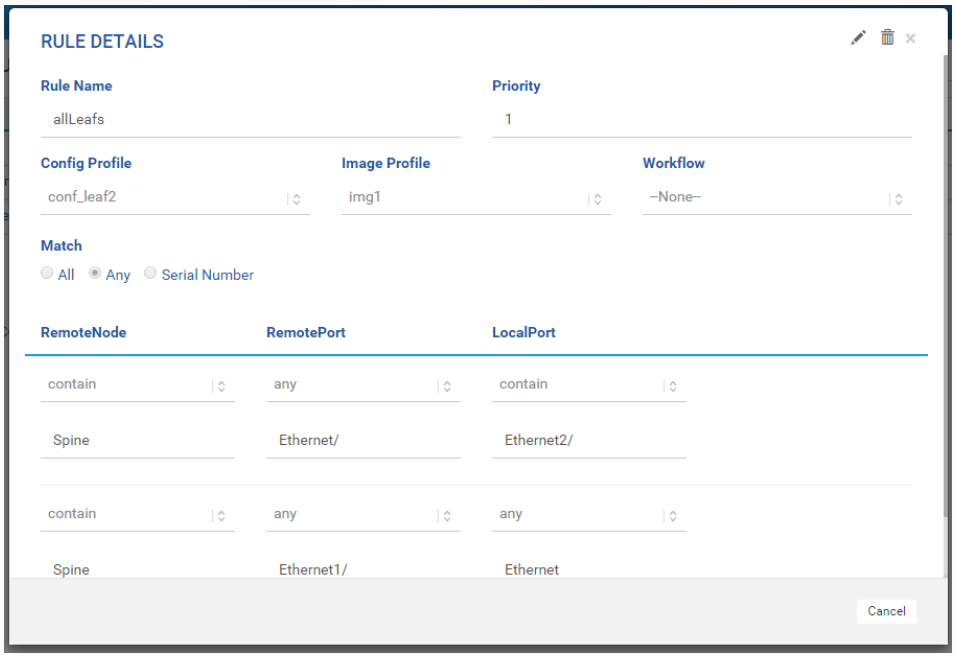
Filed	Description
Rule Name	Assign a unique name for this rule
Priority	A sequence number used in searching through discovery rules. Ignite server searches through discovery rules starting from the lowest priority value to highest priority value. First match will be used for further processing.
Config Profile	Name of CONFIG PROFILE used to build the startup configuration. Values can be filled from the drop down available in this option. Only CONFIG PROFILEs submitted, will be available for selection.

Image Profile	Name of IMAGE PROFILE with switch image details to be sent to the switch to boot. Values can be filled from the drop down available in this option.
Workflow	Additional user defined tasks to be run during POAP boot process in the switch, ***Note: Workflows are explained in Advanced Configuration section.***
Match	Select "All" or "Any" to define a discovery rule match using CDP neighbor data. "All" is used if multiple expressions are used to match and all of them should be return success. "Any" is used if multiple expressions are used to match and any of them should return success.
RemoteNode	Expression to compare and match Remote Node name.
RemotePort	Expression to compare and match Remote Port name.
LocalPort	Expression to compare and match Local Port name.
Match constructs	Following match constructs are provided to build the discovery rules: contain – contains a string defined by user e.g. contain "spine" no_contain – does not contain a string defined by user e.g. no_contain "leaf-5" match – matches a string or a regular expression no_match – does not match a string or a regular expression any – any value (***Note: when match construct is any, parameter value should also be made any***) none – no value is present

On a successful rule match, Ignite will start building the configuration file using the definition provided by administrator for *Build Configuration,* named in the rule.

In the example below a discovery rule named "`allLeafs`" is defined. The rule has two match expressions and the success criteria for the rule is "`Any`". First match expression compares the "`RemoteNode`" to "`contain`" a string "`Spine`", "`RemotePort`" to match "`any`" string and "`LocalPort`" to contain "`Ethernet2/`". Second expression, compares the "`RemoteNode`" to contain a string "`Spine`", "`RemotePort`" to contain "`Ethernet1/`" and "`LocalPort`" to contain "`Ethernet1/`". The rule is successful for a leaf switch whose uplink is "`Ethernet2/1`" or

"`Ethernet2/2`" or "`Ethernet2/*`" and is connected to spine switch named "`PodA_Spine1`" on any port. The rule is also successful for a leaf switch who has a remoted node "`PodA_Spine1`" and any of its uplinks are connected to this spine switch on port "`Ethernet1/1`" or "`Ethernet1/2`" or "`Ethernet1/*`". Here the remote node name "`PodA_Spine1`" contains the string "`Spine`" as defined in the discovery rule "`allLeafs`". On successful match, Ignite will build startup configuration using CONFIG PROFILE "`conf_leaf2`" and apply image profile "`img1`".



Setup TOPOLOGY and FABRIC

Typically a data center with multiple pods or fabrics will have many switches and defining all of these switches using DISCOVERY RULE does not scale. Startup configuration, which can be built and downloaded using DISCOVERY RULE is limited.

Topology and Fabric based discovery supports many features to create a rich set of startup configurations, customized for each switch taking into account the complete network topology.

Topology/Fabric information in data center network is represented using:

- Topology tiers – Leaf, Spine, Core router, Border router
- Topology tier defaults – switch models used in different tiers, image profiles used in different tiers, type of links used to connect switches between tiers, number of links for connection between switches
- Number of switches in each tier
- switch name – host name association
- Connectivity between switches

TOPOLOGY is a network template. FABRIC is an actual instance of a network using a TOPOLOGY. Many FABRICs can be defined using the same topology. A FABRIC can initially be instantiated from a TOPOLOGY and later evolve by adding or deleting switches, adding or deleting links in any tier. FABRIC represents real implementation of a data center network and exists independent of TOPOLOGY. Changes made to TOPOLOGY are not propagated to FABRICs instantiated from this topology.

Once the topology with its related components is defined, administrator can describe the fabrics deployed in the data center using this topology. Many fabrics will use the same topology.

Following topological information available in a FABRIC can be used in CONFIGLETs to build customized configurations for a switch in the FABRIC using a CONFIG_PROFILE.

Parameter	Value
HOST_NAME	Each switch in a FABRIC has a unique name assigned by the user or auto generated by Ignite. Once a switch is identified from its CDP neighbors and association of that switch with a specific FABRIC is made, value of HOST_NAME is taken from the name assigned to this switch in that FABRIC instance.
VPC_PEER_LINK_IF_NAMES	Names of VPC peer link Interfaces

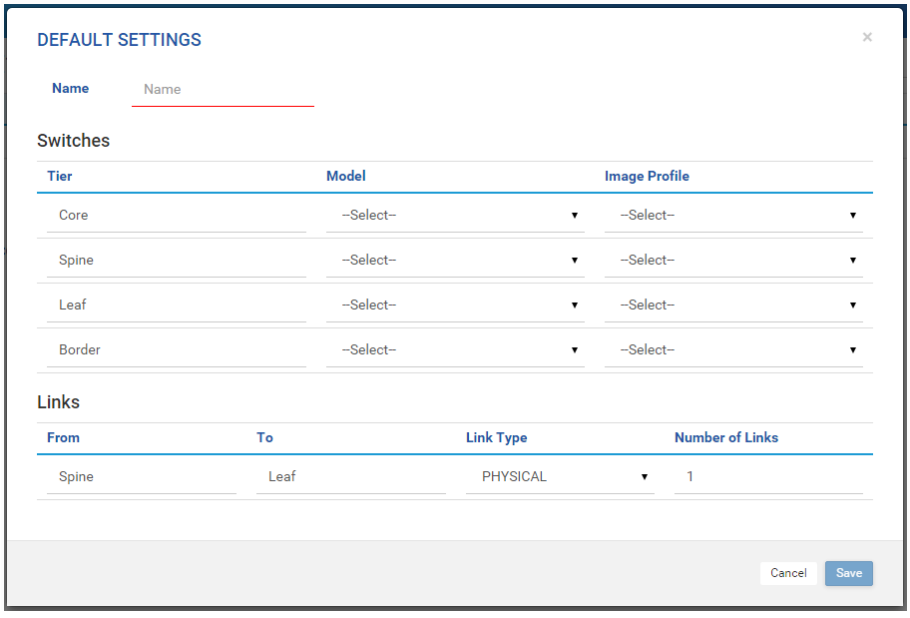
VPC_PEER_DST	IP Address of the VPC neighbor to which this switch is connected to
VPC_PEER_SRC	IP address assigned to this switch
TRUNK_PORTS	Interfaces used between Spine and Leaf. Spine switch will refer to interface names used to connect to Leaf switch and Leaf switch will refer to interface names used to connect to Spine Switch.

Creating a new TOPOLOGY

Select “TOPOLOGY” option in “FABRICS” tab. List of topologies defined in the system is displayed.

A new TOPOLOGY is created by clicking “+Add” button in TOPOLOGY list view.

DEFAULT SETTINGS form will be displayed. Fill the form fields and click SAVE to continue building TOPOLOGY.

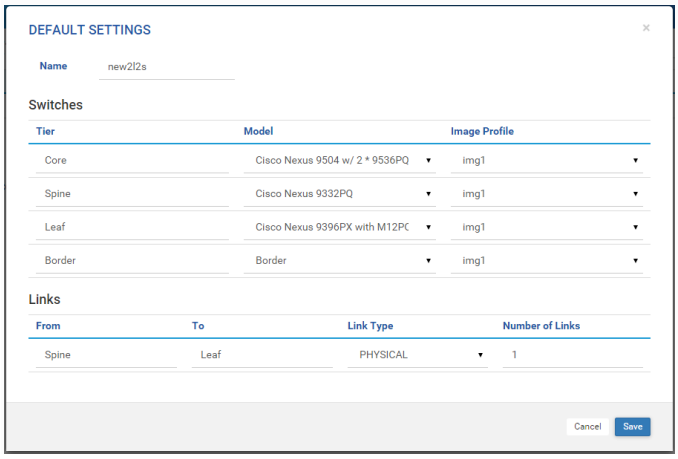


Topology DEFAULT SETTINGS form has the following fields:

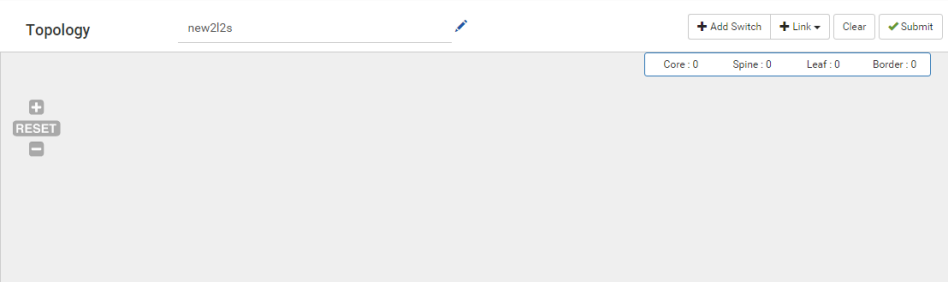
Field	Description
NAME	Topology name
Switches- Tiers	Core, Spine, Leaf and Border are available.
Model	Switch Model used in each tier. If a specific switch or switches in a tier use different model, it can be changed once switch is added to topology.
Image Profile	Default Image profile used in each tier
Link Type	Type of link connecting a switch from Leaf tier to its neighbor in Spine Tier. Options are PHYSICAL or PORT_CHANNEL. ***Note: This information is used in building startup configurations through FABRIC_PROFILE, which is explained in advanced configuration section.***
Links	Number links connecting a switch from Leaf tier to its neighbor in Spine Tier.

Example showing TOPOLOGY DEFAULT SETTINGs.

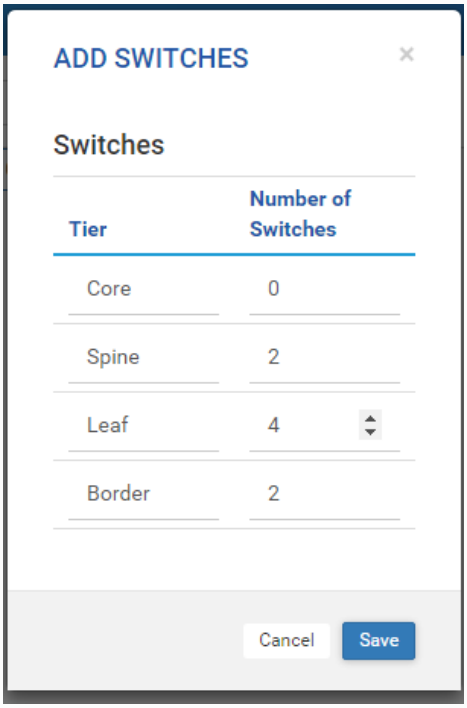
Note: all tier defaults should be filled to save the settings.



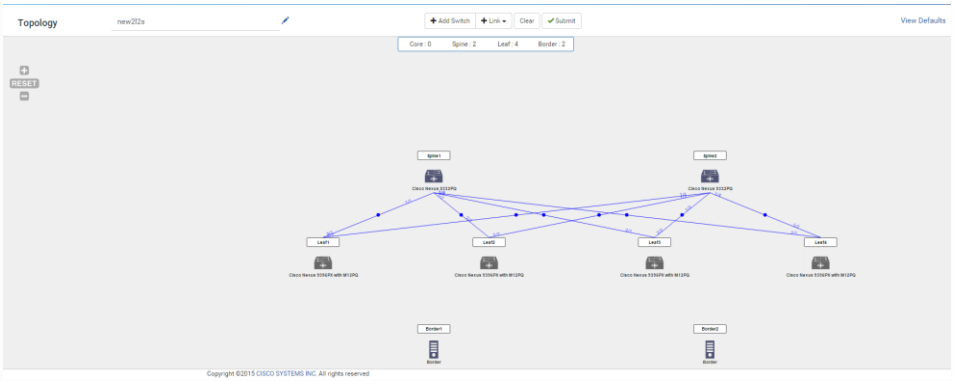
Once TOPOLOGY DEFAULT SETTINGS is saved, a TOPOLOGY design pane will be displayed. Use this design pane to define your network topology.



Click “+Add Switch” to add new switches to topology design pane.



Enter number of switches you want to add and click Save.



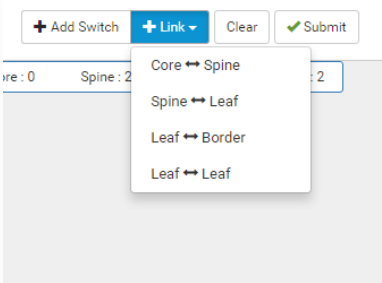
Topology diagram showing the number of switches and links connecting Leaf and Spine tier switches is displayed.

Leaf to Spine connectivity will use number of links as specified in TOPOLOGY DEFAULT SETTINGS. Port number of the links will be determined from the switch model used in each tier. Leaf tier switches will use the ports defined as uplinks in the switch model selected for Leaf tier switch. Spine tier switches will use the ports defined as downlinks in the switch model used. Port numbers are automatically generated.

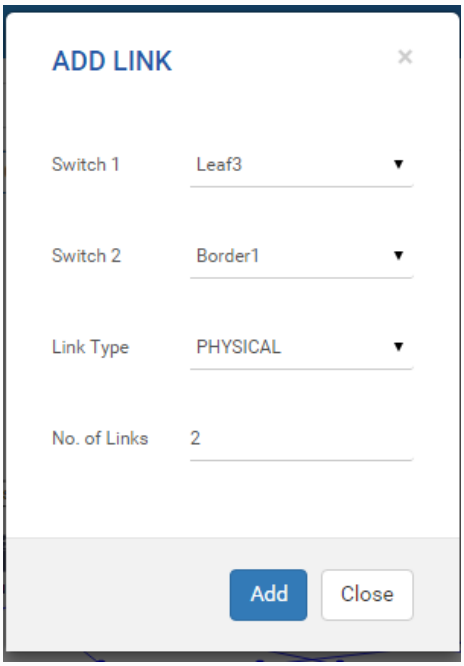
Note: In TOPOLOGY, links between LEAF and SPINE tiers which are generated automatically cannot be modified or deleted. Changes to LEAF and SPINE connectivity can be done in FABRIC. Links between Leaf/spine and border/core router are not created automatically. These links should be added manually.

Adding topology links between LEAF/SPINE switches and BORDER/CORE routers

To add link between LEAF and BORDER tier, select drop down option from “+Link” button, TOPOLOGY design pane.

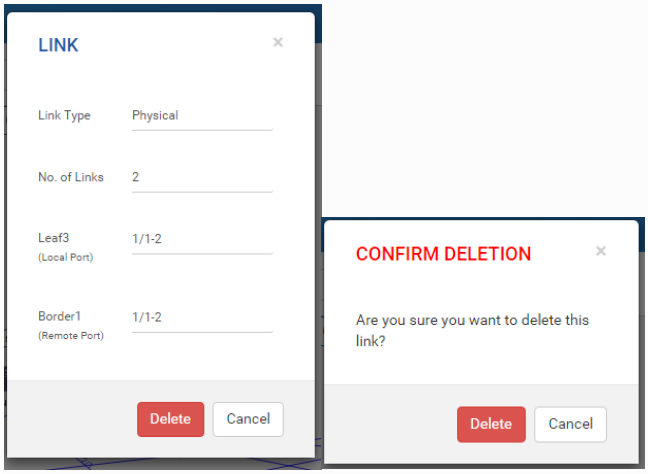


ADD LINK form will be displayed. Fill the form with details and save to connect a leaf switch with Border router.



Port numbers are generated by Ignite from switch models used in the respective tiers using uplink and downlink definitions and the next available port in the switch.

To view link details between switches, click on the circular blob (- ■ -) on the link connecting the two switches.



Links added using “+Link” can be deleted after selecting the link details and confirming deletion.

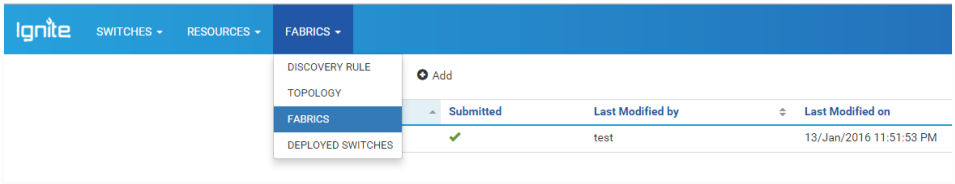
Submit TOPOLOGY

Once TOPOLOGY is fully defined, submit it by clicking “Submit” button on the TOPOLOGY design pane. Once submitted, TOPOLOGY can be used to create a FABRIC.

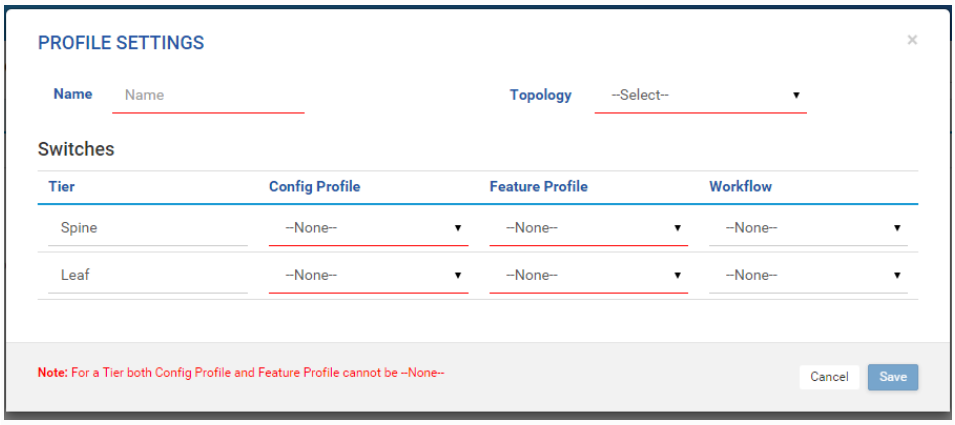
FABRIC

FABRIC represents a data center network details. FABRIC is built from a TOPOLOGY, but can be modified to match the real network connectivity.

To create a new FABRIC, select “FABRICS” option from “FABRICS” tab.



From the list view of FABRICS, click “+Add” button to create a new FABRIC. PROFILE SETTINGS form will get displayed. Fill details on this form and click “Save” to continue with FABRIC design.

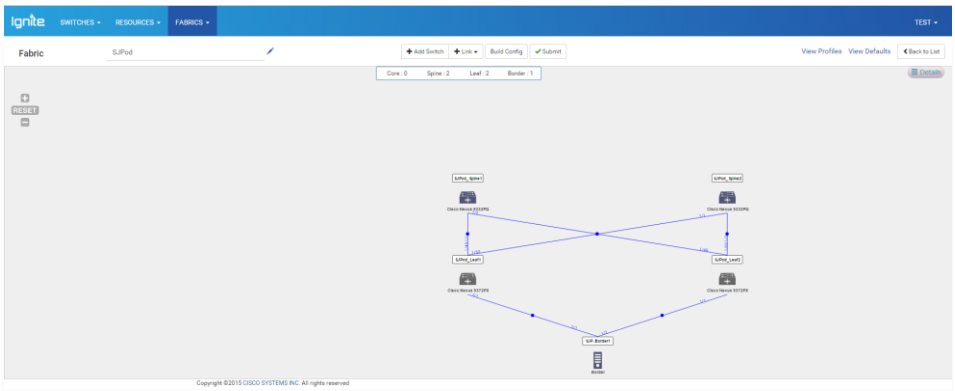


In the PROFILE SETTINGS form, following details are required.

Field	Description
NAME	Name of the FABRIC. ***Note: This name will be appended to the switch name created by Ignite for all switches present in this fabric. These names can be changed later as required.*** For example: if FABRIC name is "SJCPod", then all spine switches will be named as "SJCPod_Spine1", "SJCPod_Spine2" etc. Similarly for Leaf switches.
Topology	Select a TOPOLOGY to use to build this FABRIC
Tier	Spine and Leaf Tier default profiles can be displayed.
Config Profile	Default CONFIG PROFILE used to build the startup configuration for the switches in this tier. This profile can be individually overridden for each switch, once the FABRIC is created.
Feature Profile	Default FEATURE PROFILE used to build advanced configurations for the switches in this tier. This profile can be individually overridden for each switch, once the FABRIC is created.
Workfflow	Default "Workflow". Workflows are tasks to be performed during the POAP boot process in addition to switch image download and startup configuration setup. Typical tasks would be to load other packages like , Puppet agents, Monitoring scripts etc.

Note: Either Config Profile or Feature Profile is needed to save PROFILE SETTINGS.

Once the PROFILE SETTINGS is saved, FABRIC DESIGN pane will get displayed with network topology.



To view network topology connection details, click on “Details” button.

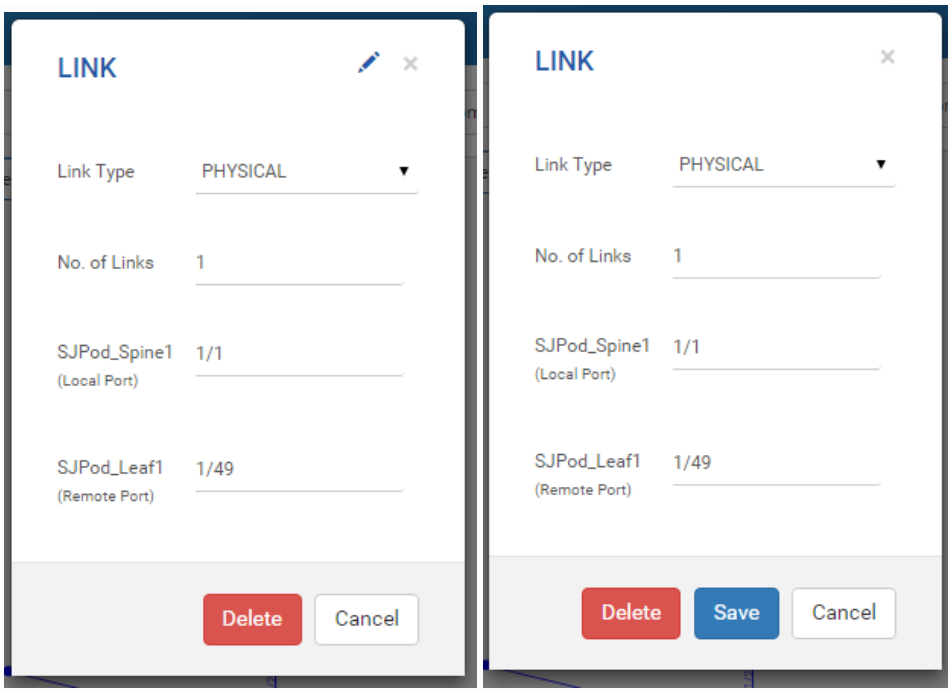
DETAILS

Search

#	Node	Local Port	Remote Node	Remote Port	Action
1	SJPod_Spine1	1/1	SJPod_Leaf1	1/49	
2	SJPod_Spine2	1/1	SJPod_Leaf1	1/50	
3	SJPod_Spine1	1/2	SJPod_Leaf2	1/49	
4	SJPod_Spine2	1/2	SJPod_Leaf2	1/50	
5	SJPod_Leaf1	1/1	SJPod_Border1	1/1	
6	SJPod_Leaf2	1/1	SJPod_Border1	1/2	

To change the connectivity details from DETAILS view, you can click on VIEW icon from Action column.

Link details will be displayed. Select EDIT to make changes to any of the link properties and save. Links can also be deleted using DELETE option.

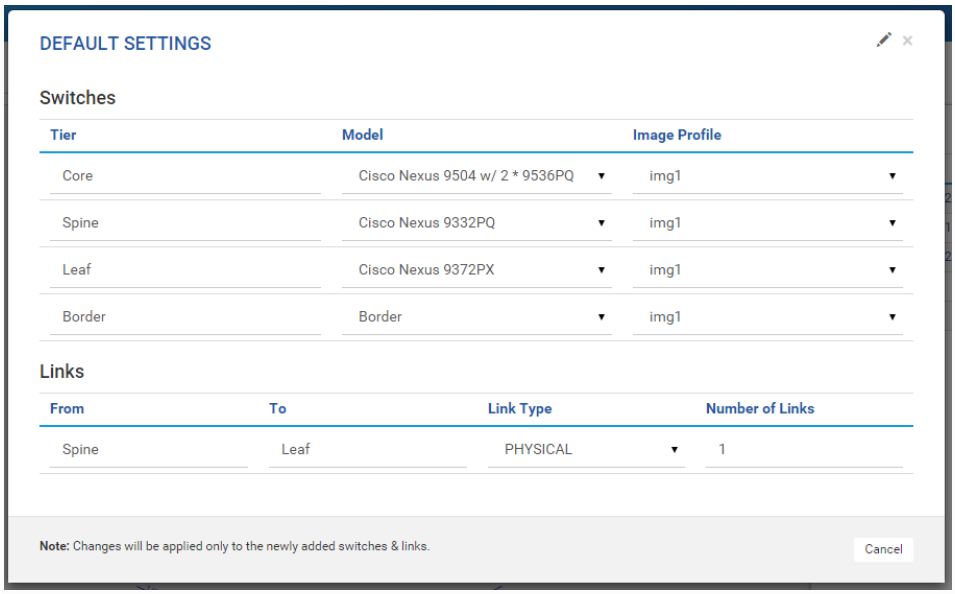


FABRIC will inherit DEFAULT SETTINGS from the TOPOLOGY used while creating the FABRIC. Switch model, image profile, links used to connect between leaf and spine tiers are defined in the DEFAULT SETTINGS.

Edit DEFAULT SETTINGS and PROFILE SETTINGS

DEFAULT SETTINGS and PROFILE SETTINGS can be modified.

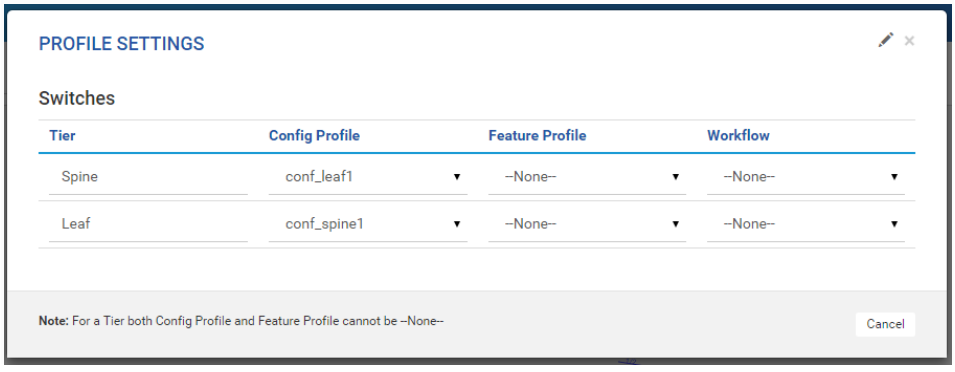
To view or edit DEFAULT SETTINGS, click on “View Defaults” button.



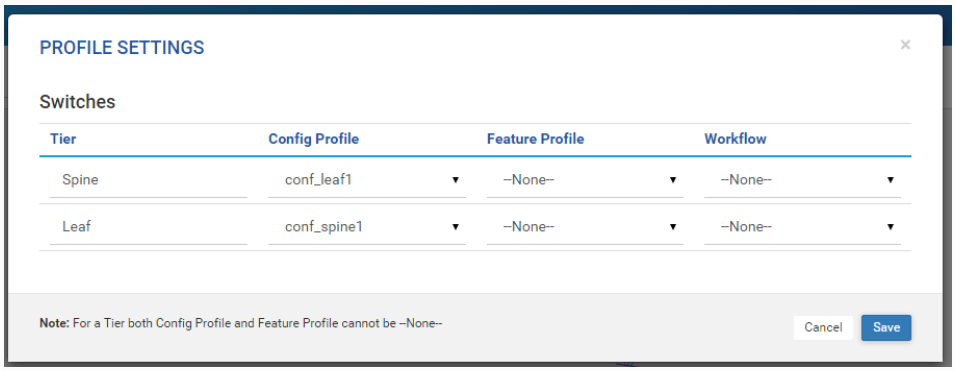
Model, Image Profile and Links properties can be changed.

Note: any changes in default settings will be applicable only when a new switch is added to the FABRIC. Switches already created from TOPOLOGY DEFAULT SETTINGS will still retain the original values. Changes to these elements can be done manually.

To view or modify PROFILE SETTINGS click on “View Profiles” button.



Tier level profile settings are displayed. Config Profile, Featuer Profile and Workflow cab be modified, by clicking edit icon, changing values and finally saving it by clicking “Save” button.

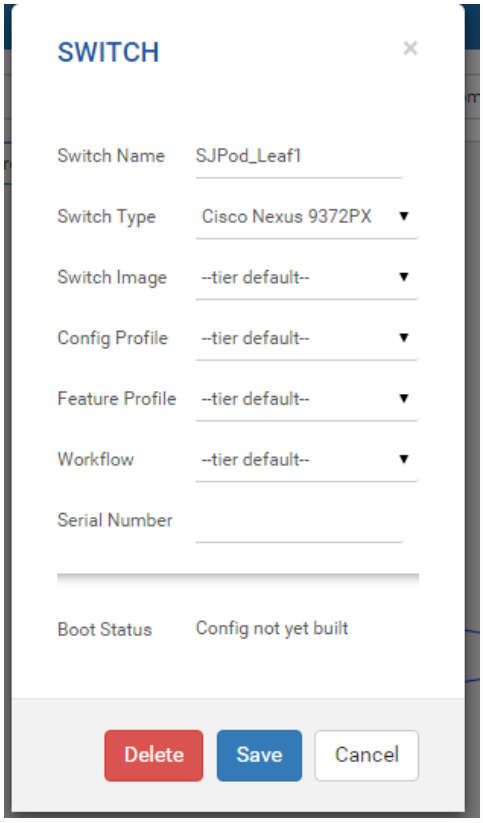


PROFILE SETTINGS changes take effect when the next switch from this FABRIC requests POAP boot.

Viewing and editing switch DEFAULT and PROFILE settings

Initially when FABRIC is instantiated, all switches in a tier are assigned the same defaults and profiles from tier level settings. Individual switches can have default and profile settings different from the tier level defaults.

To modify the settings, in FABRIC DESIGN pane, click on the switch icon for the desired switch. All settings associated with switch will be displayed. Select edit icon, modify values and save. Modified values take effect, when switch requests POAP boot next.



Note: clicking DELETE will delete the complete switch with its associated links from FABRIC.

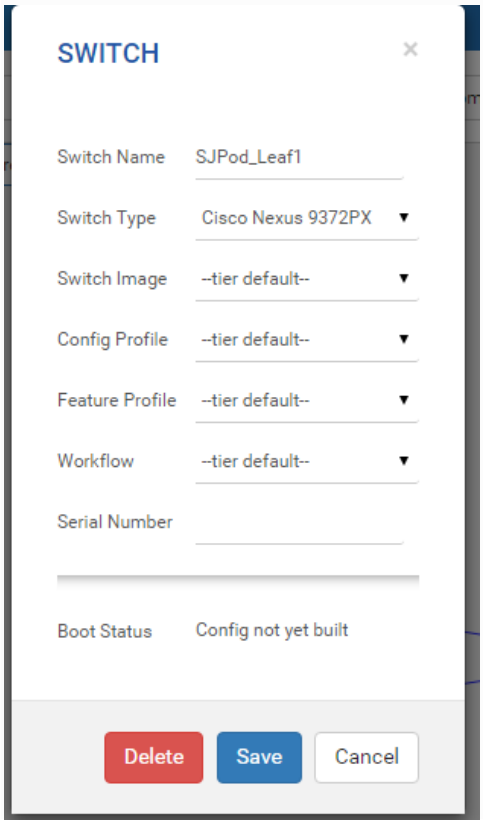
Adding Serial Number of a switch in FABRIC

The purpose of defining a TOPOLOGY/FABRIC is for the ability to build start up configuration for switches intelligently based on their location in FABRIC, its neighbor and connectivity between its neighbors.

When a FABRIC is being installed new and when the first switch requests POAP boot configuration, it will not be able to identify its FABRIC neighbors as they are not powered on and booted. In this scenario, it is necessary to have at least one switch in

LAEAF and SPINE tier in a FABRIC be identified using Serial Number and those switch powered ON and booted using POAP first, before other switches are powered ON and booted. When this Switch boots up, it does not send any FABRIC neighbors, but will send its Serial Number in the boot request. This serial number is searched first in the FABRIC database to see if it belongs to any of the FABRIC and if so Ignite recognizes its identity from FABRIC TOPOLOGY. Only if serial number is not found in FABRIC, Ignite will search DISCOVERY RULEs to see if the switch can be identified from these match rules.

Switch Serial Number can be added by selecting a switch from FABRIC DESIGN pane (click on switch name or switch icon). Use edit icon to enter edit view of switch details and add a Serial Number.



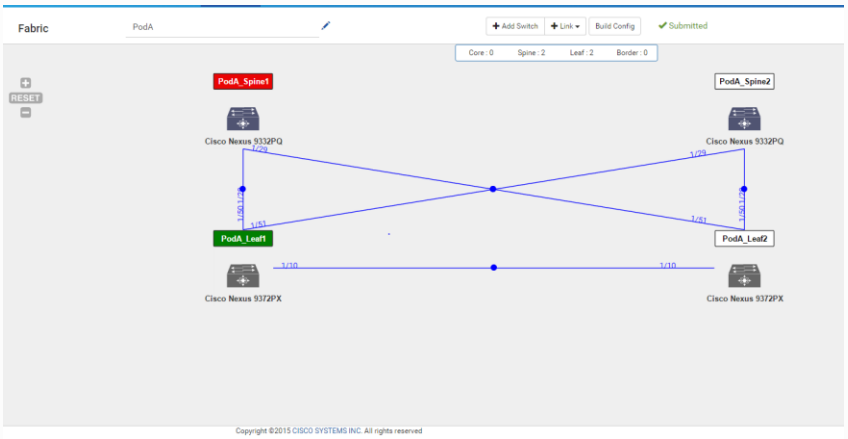
Note: Serial Number should be unique. This serial number should not be defined in any other DISCOVERY RULE or FABRIC.

Error message indicating conflict of use will be displayed, if a Serial Number conflict is identified.

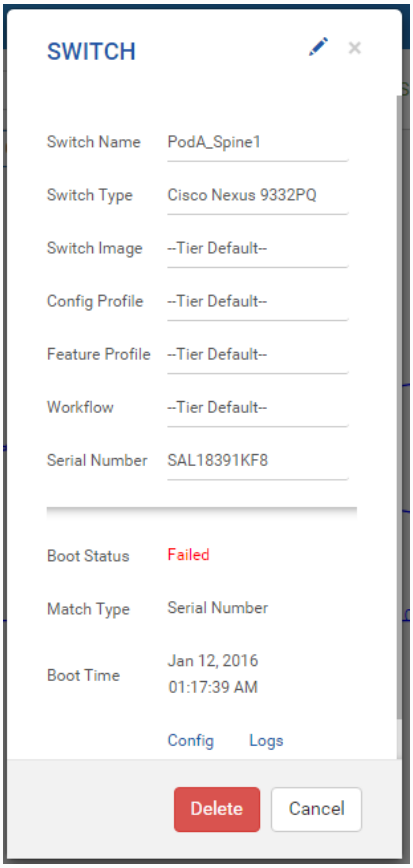
Deployed switch status

FABRIC DESIGN pane provides a status view of switches booted through POAP requests. In FABRIC DESIGN pane, switch name box is highlighted with colors to indicate boot status of the switches.

Green – successfully booted
Red – boot failure
Orange – boot in progress



Clicking on switch name or switch icon displays switch details,



Switch details will display

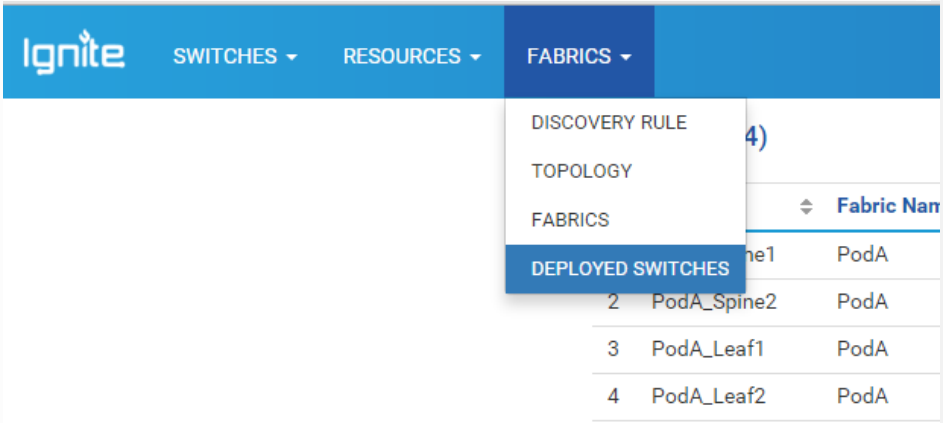
- Boot Status – success or failure
- Match Type - rule used to discover the switch
- Boot Time: time switch requested boot
- Config link - link to view startup configuration downloaded to the switch,
- Logs link - link to view the bootstrap script log (ignite_poap.py log)during POAP process

Example Logs display:

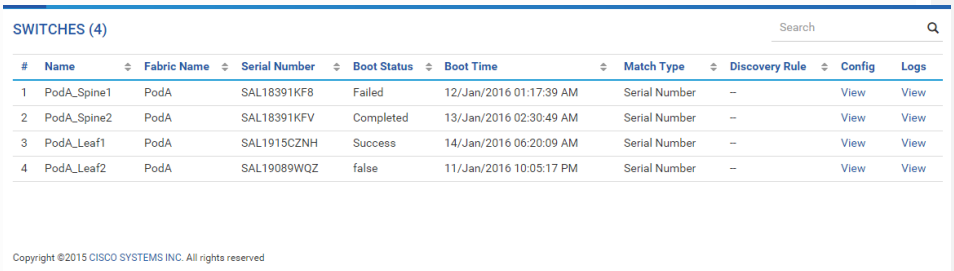
Example Config display:

While FABRIC DESIGN pane displays boot status of switches in FABRIC,

“Deployed Switches” shows status of switches which were booted through Fabric discovery. To view status of all switches booted through FABRIC or DISCOVERY RULES, select DEPLOYED SWITCHES to view the list.



List view of deployed switches:



List view displays, list of switches brought up so far, boot time for each switch, configuration which was downloaded etc. is displayed. Clicking on Config link, will display start up configuration downloaded to the switch. Logs link will display log from bootstrap script (ignite_poap.py).

Advanced bootstrap configuration – user defined POAP tasks

Ignite provides you flexibility to add custom scripts to be run in the switch during POAP process. This is achieved through client tasks which can be added to the POAP script.

Ignite_poap.py the bootstrap created by ignite, by default does the following:

1. Setup environment to run the POAP process
2. Creates a list of interfaces in the switch
3. Runs “no shut” on these interfaces
4. Gets CDP neighbor information
5. Prepares a request containing switch serial number, neighbor details and sends an HTTP request to Ignite server
6. Waits for response from Ignite server
7. Ignite server identifies the switch in its database, builds startup configuration file and image profile. Then it creates a list of tasks to be run by ignite_poap.py in the switch and sends this list to switch.
8. For each task, task list contains the following information:
    a. Task handler and function name – task handler is a python script file (path and filename); function name is a function in the python script, which should be executed to run the task handler script
    b. Location – IP address of the server where the task handler script file is located, transfer protocol to be used and credentials required
    c. Parameters – while initiating function in handler script, arguments passed to the function
9. Ignite will create a task list by default with 2 tasks, “bootstrap_image” and “bootstrap_config”. Handlers for these are pre-packaged in Ignite and all the details required in the task list are filled by Ignite.
10. After running these task handlers and updating running /scheduled configs, switch reboots.

In addition to system image and startup configuration, data center switches today also require custom configurations based on their data center environment and management needs. Some examples include:

- Puppet agent installation
- Chef client installation
- Nexus Application Development SDK installation
- 3rd party native application installation
- 3rd Monitoring/Management systems plugins installation
- Scripts and packages needed for operation

These additional configurations/package installation can be done during the POAP process in order to accelerate operational readiness of FABRIC infrastructure.

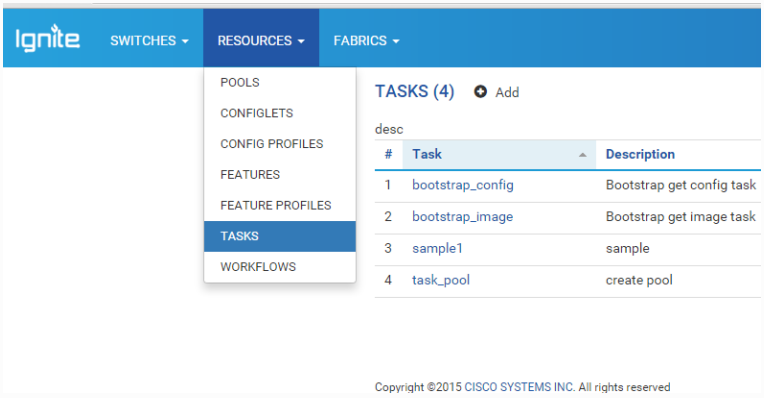
To run client tasks during POAP process in addition to the standard image and configuration downloads, user should create a task and then add this task to a workflow. This workflow can be cloned from “bootstrap_wf”, which is predefined by Ignite. After cloning the new workflow add the task. This workflow can be assigned to a DISCOVERY RULE, PROFILE SETTINGS of a FABRIC or PROFILE SETTINGS of a switch in FABRIC.

Adding a new client task

Prior to adding the client task to Ignite for use in POAP, following steps should be completed:

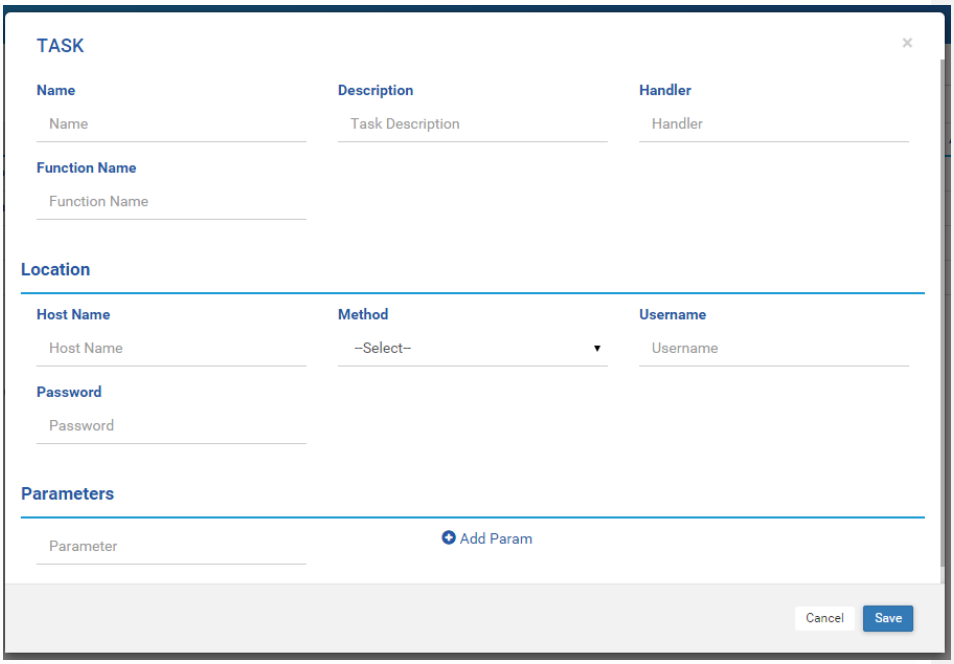
1. Prepare client task handler – task handler is a python script, which will be loaded and run on the switch during POAP process. This script file should be created and tested to run on the switch.
2. Identify the function to run the script along with its parameters. Task handler is loaded to the switch and run as a module. Task handler gets control through the function call. When the function is called, parameters will be assigned a fabric/switch specific value provided by user.
3. Copy the task handler script to a server – task handler should be available in a server directory which can be accessed by switch during POAP process. Bootstrap script from switch downloads task handler from this directory. Task handler script directory could reside in Ignite server, but is not managed or maintained by Ignite application.

Once the task handler is ready to be added to Ignite server for use in FABRIC configuration, select “TASKS” option from “RESOURCES” tab in the top menu.



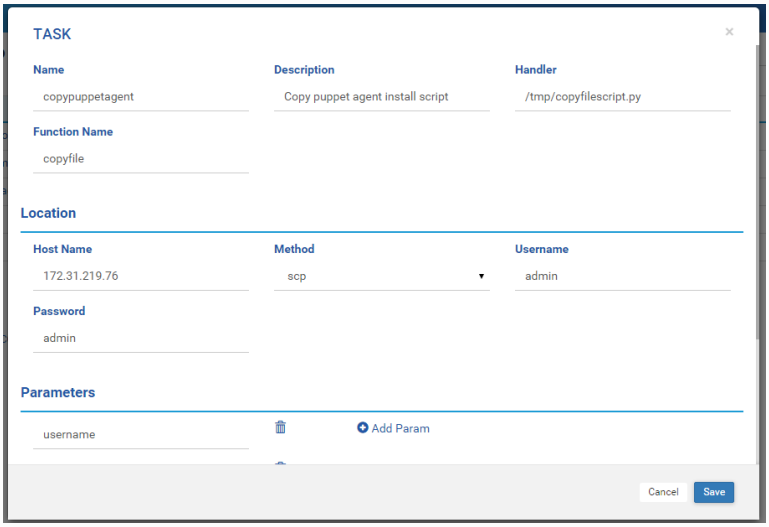
From task list view click “+Add” to create a new task,

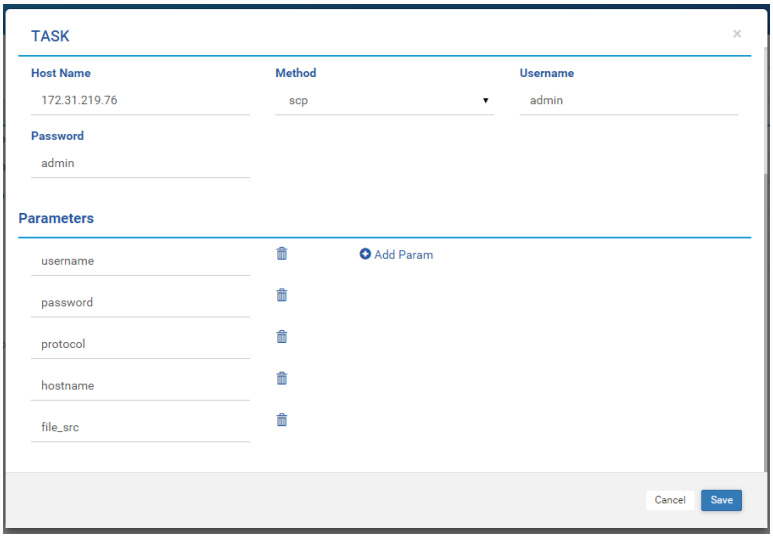
Note: task list by default, includes bootstrap_config , bootstrap_image. These Ignite generated client tasks which are run during POAP process. These tasks cannot be edited or deleted.



Fill in the task details in TASK form and save. Task is now available for use. Fields in TASK form are:

Field	Description
Name	Task Name. This name is referred in WORKFLOW.
Description	Brief functionality described in English for help and reference.
Handler	Filename with path required to download this task.
Location – Host Name	IP address of the server where the task handler resides.
Method	Transfer protocol method used to transfer the file.
Username	Username required to access the server to download
Password	Password required to access the server to download
Parameters	List of one or more parameter names. These names should be same as that is present in the function description defined in task handler. Add one or more parameters as required.

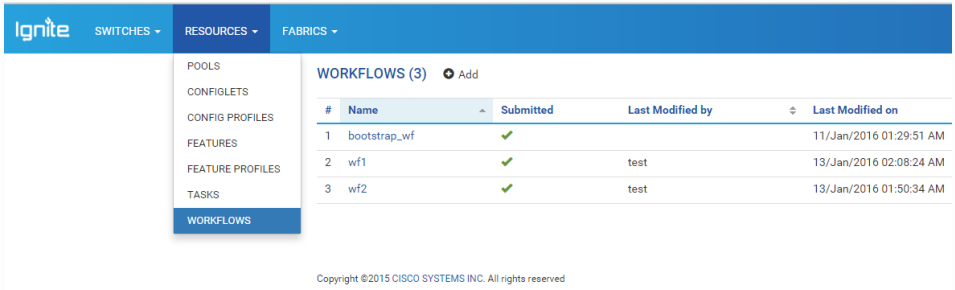




Once the task is created, this can be added to a workflow.

Creating a new workflow

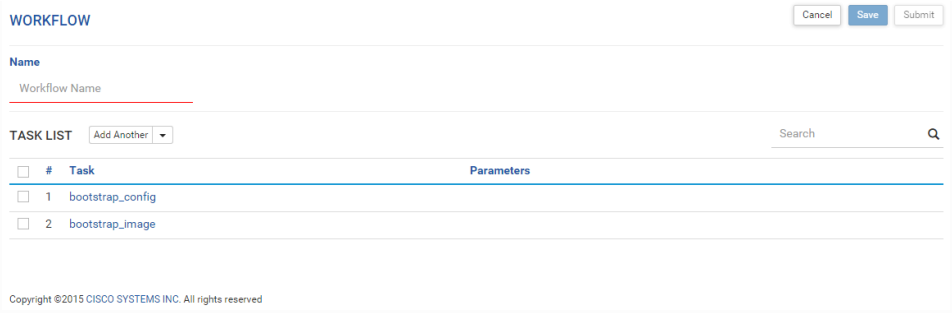
Select “WORKFLOWS” option in “RESOURCES” tab from top menu. List view with available workflows is displayed.



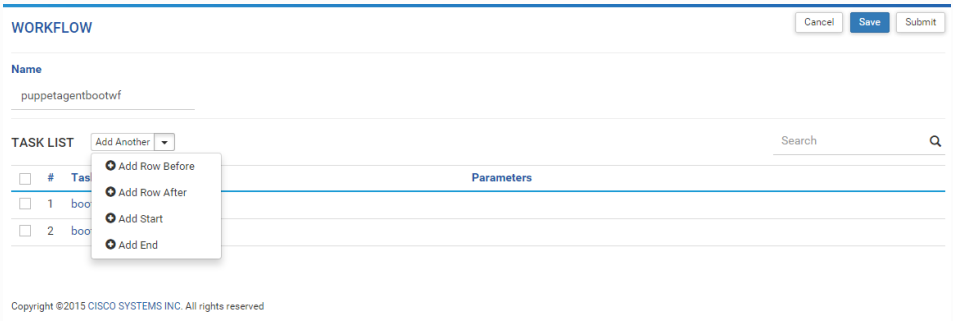
Clone “bootstrap_wf” from the list. This will create a new workflow with bootstrap_image and booststrap_config tasks already added to the workflow. The new task which you want to add can be added before or after these pre-defined tasks in the workflow.

Alternatively click “+Add” to create a new workflow. You can then manually add the pre-defined bootstrap tasks “bootstrap_image” and “bootstrap_config” to the workflow in their desired positions along with other tasks which you want to add.

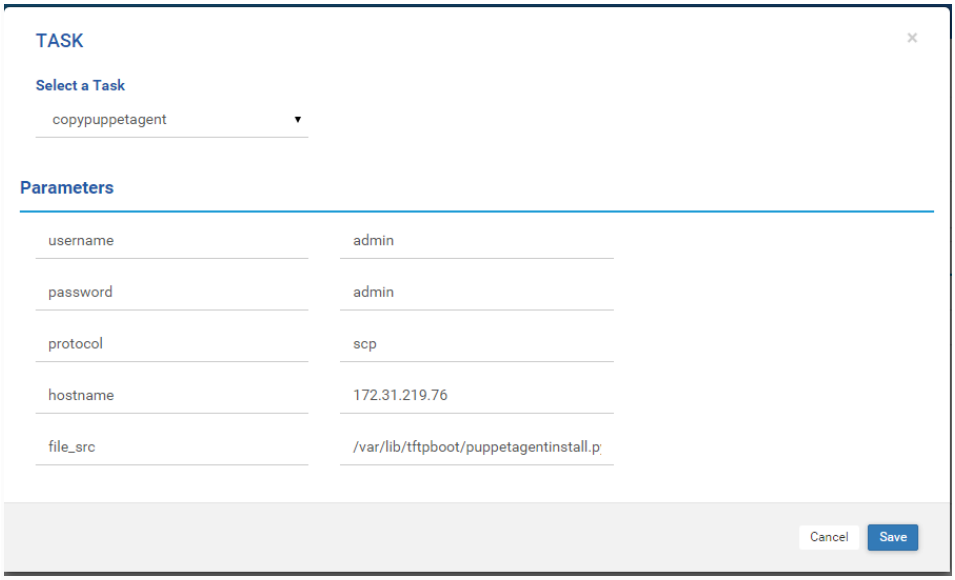
Note: In order for Ignite to download startup configuration and switch image from the desired image profile, it is mandatory to have “bootstrap_config” and “bootstrap_image” tasks included in a workflow assigned to a switch through DISCOVERY RULE or FABRIC. In the absence of these tasks, unless client tasks are able to perform the same function as that of the pre-defined tasks, switch may not boot with the desired configuration.



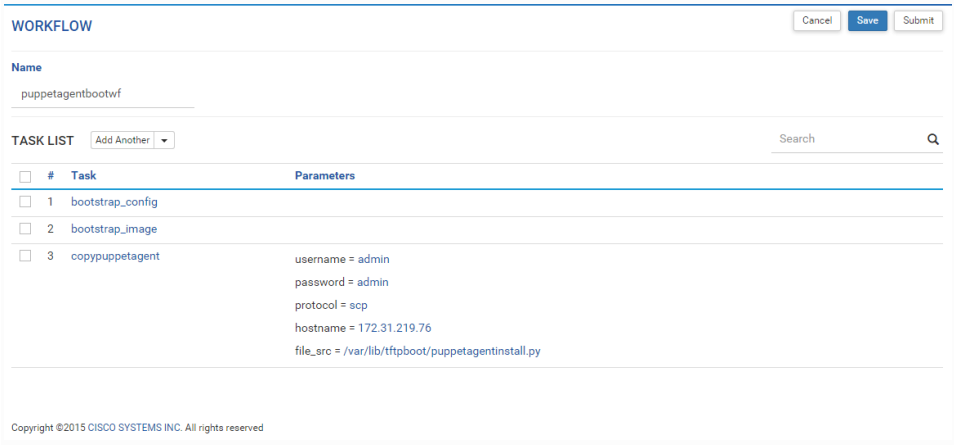
Modify the cloned workflow by adding tasks in the desired sequence with respect to pre-defined bootstrap scripts.



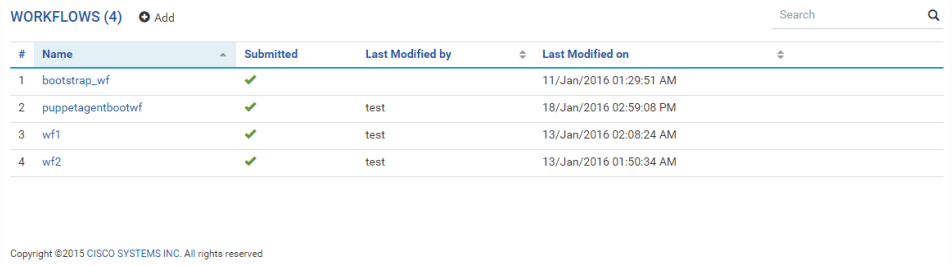
Once the new task is selected, in the TASK form, fill in all the parameter values required by the task function and save.



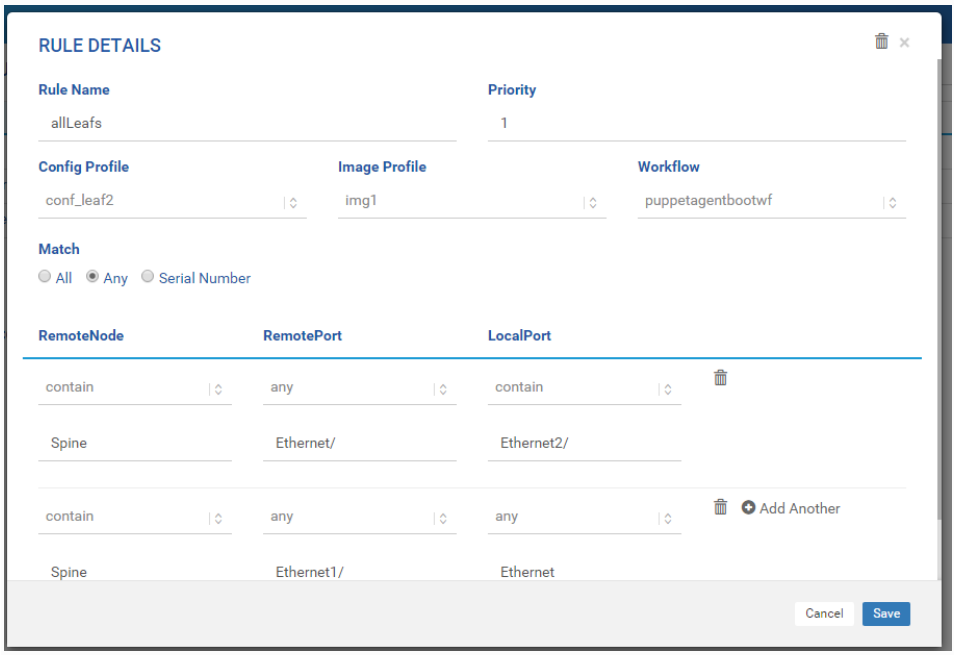
After adding all tasks click “SUBMIT” to submit the workflow.



Once submitted, workflow will be available to be applied in FABRIC / DISCOVERY RULE.

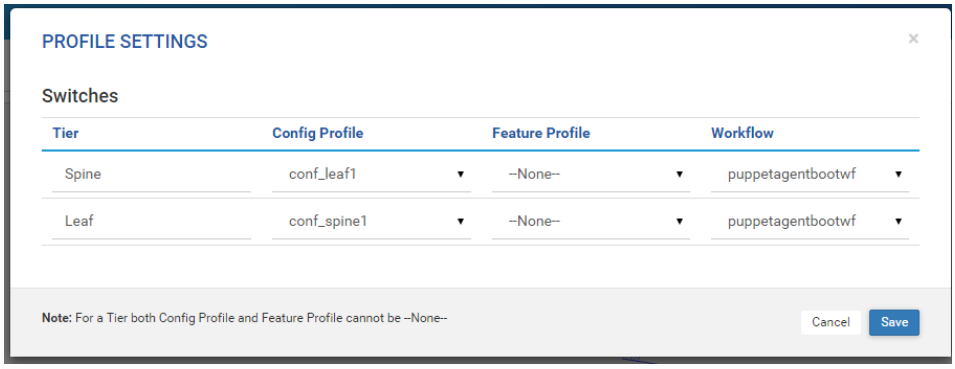


To apply workflow to DISCOVERY RULE, select the desired DISCOVERY RULE, edit, add workflow and save.

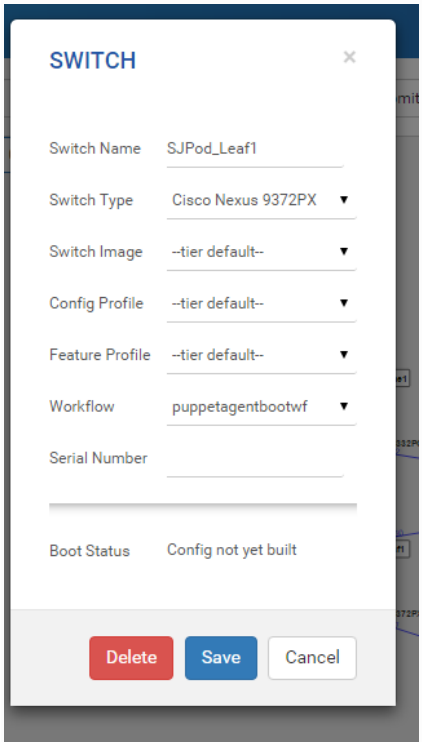


Note: workflow will be applied to all with switches discovered through this rule.

To apply workflow to FABRIC, select FABRIC, view PROFILE SETTINGS, select edit, add workflow and save.



To apply workflow to a selected switch in FABRIC. Select FABRIC and view the desired fabric in FABRIC DESIGN pane. Select a switch by clicking on the switch name or icon, switch details are shown, click edit and modify workflow profile and save.



FEATURES and FEATURE PROFILES

FEATURES and FEATURE PROFILES are constructs which enables Ignite server to interface with AutoNetkit. Autonetkit is a network management tool, which automatically generates network configurations from a topology. It extracts the topology information and builds configurations required to enable BGP, OSPF protocols between network elements. It also has the ability to allocate IP addresses required for such configurations. It is a highly flexible and configurable tool for building network configurations for mutli-device, multi-vendor networks.

Since Ignite supports a topology based network discovery and configuration generation, it could easily leverage AutoNetkit capabilities to automatically generate complex configurations required for overlay and underlay networks in a data center. Ignite uses a modified version of AutonNetkit which allows passing the user defined topology and configuration requirements in a JSON format to build necessary configurations.

Through FEATURES and FEATURE PROFILES, Ignite can generate:

1. Port-channel configurations for links between tier switches
2. Virtual Port Channel domain, VPC peer-links, VPC peer keep-alive links
3. BGP
4. OSPF
5. EIGRP
6. Infrastructure VXLAN – multicast, BGP-evpn,
7. SNMP
8. NTP

FEATURES are AutoNetkit JSONs which specify a method to communicate certain parameters and/or inputs required by ANK to generate a configuration.

As an example JSON, to configure bgp in network elements an ASN number is associated with every switch involved in bgp routing. Based on the topology adjacencies and ASN number, AutoNetkit generates the appropriate configurations.

Commented [s1]: It extracts the topology information and feature profile information to build configurations for various network protocols.

Commented [s2]: Feature profile is not required for PC and VPC. So we can say that “Major protocols for which Ignite can auto-generate configurations are listed below: ”

```
"bgp":
        {
           "asn":"$$ASN_NUMBER$$",
           “route_reflector”:$$ROUTE_REFLECTOR_ENABLED$$
        }
```

Once this FEATURES JSON for bgp is created, it now needs to be associated with each switch in the topology which will run BGP. Associating or applying a FEATURE PROFILE to a switch or a set of switches is accomplished by building a FEATURE PROFILE which contains a number of FEATURES and then associating FEATURE PROFILE with the required switches in FABRICS view.

A feature profile can be created and applied on across the fabric. Right now only two such profiles is supported (introduced in 1.7):

1) Global_cfg

```
“global_cfg”:{
“pc_only”:$$PORT_CHANNEL_ONLY_ENABLED$$,
“igp_enabled”:$$IGP_ENABLED$$,
“bgp_enabled”:$$BGP_ENABLED$$,
“enabled_routing”:$$ROUTING_ENABLED$$,
“infra_block”:”$$INFRA_BLOCK$$”,
“loopback_block”:”$$LOOPBACK_BLOCK$$”
}
```

If “pc_only” is set to true then only port channel or/and VPC configuration will be generated with ip address assignment or any routing protocol configuration. If it is set to false then other configurations such as interface config with ip address, ospf, bgp etc will be generated.

Igp is disabled by default so no igp configuration will be generated by default. If “igp_enabled” is set to true then igp configuration will be generated for all the switches with IGP protocol being “ospf”. “igp.json” profile can be used to disable/enable igp on a particular switch or use some other igp(ex: eigrp, isis, rip).

Bgp is disabled by default so no bgp configuration will be generated by default. If “bgp_enabled” is set to true then bgp configuration will be generated for all the

Commented [s3]: Route_reflector option is added in 1.7. if we set route_reflector_enabled to True then the switch on which this profile is applied will act as route reflector.

Commented [s4]: It doesn’’t needs to be applied on all switches. Only the switch which deviates from default need to have this profile. Default is: ASN will be 1 for the switch and if the switch type is spine then it will act as route reflector.

switches. Using bgp.json profile on individual switches has been explained earlier only.

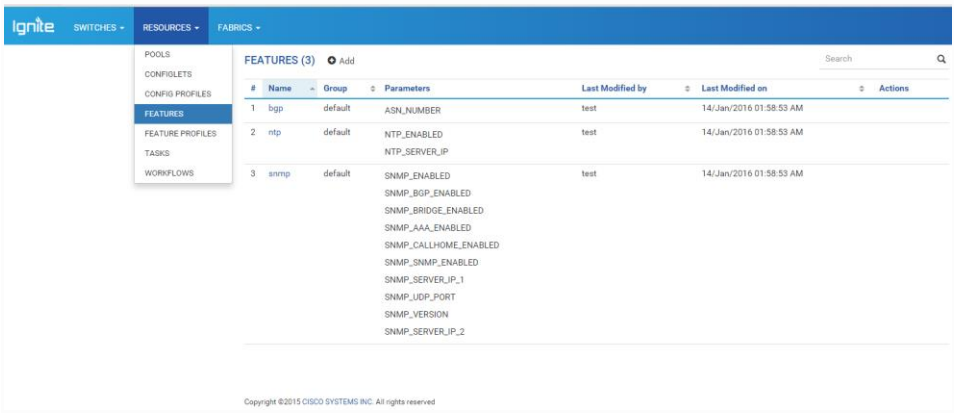
"enable_routing" when set to false disables all routing protocols.
"infra_block" can be used to assign interface ip addresses from a certain block. Similar is the case for "loopback_block"

2) Vxlan_global: This is used to configure global parameters of Vxlan. Profile json is self explanatory.

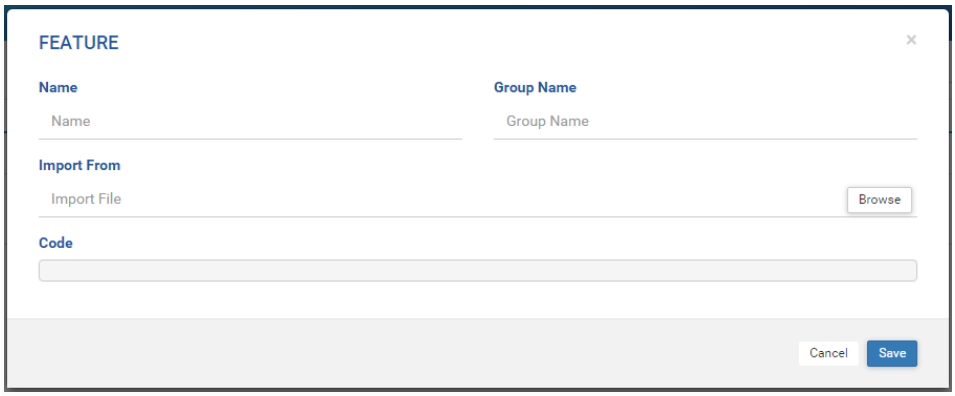
Commented [s5]: Added in 1.7

Adding a new FEATURE

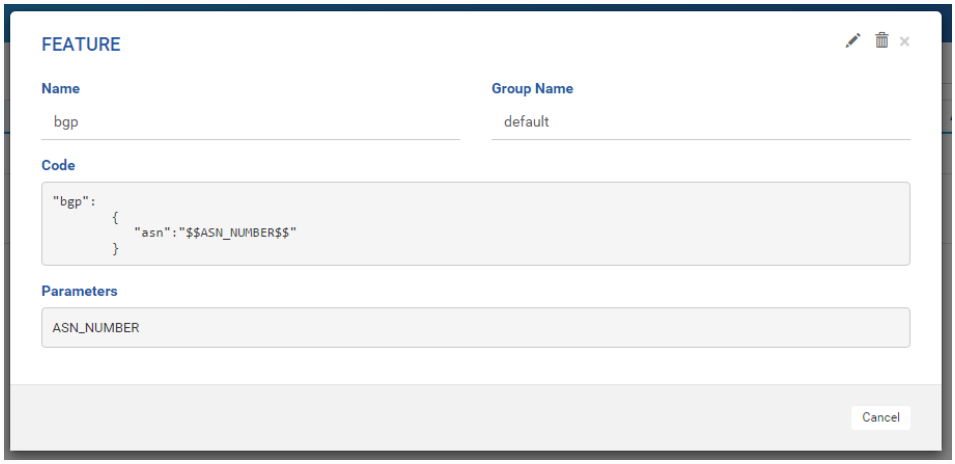
To add a new FEATURE, select "FEATURES" option in "RESOURCES" tab. List view of available FEATURES is displayed.



Add new FEATURE by clicking "+Add", fill out details.



FEATURE JSON is imported from user system and assigned a name and group name. Group name is used to categorize FEATURES (similar to file directories). Similar to CONFIGLETS, FEATURE JSONS also identify parameters using delimiter “$$”. For example a string $$ASN_NUMBER$$ in a FEATURE JSON will be automatically identified by Ignite as a parameter required by this FEATURE and saved accordingly.

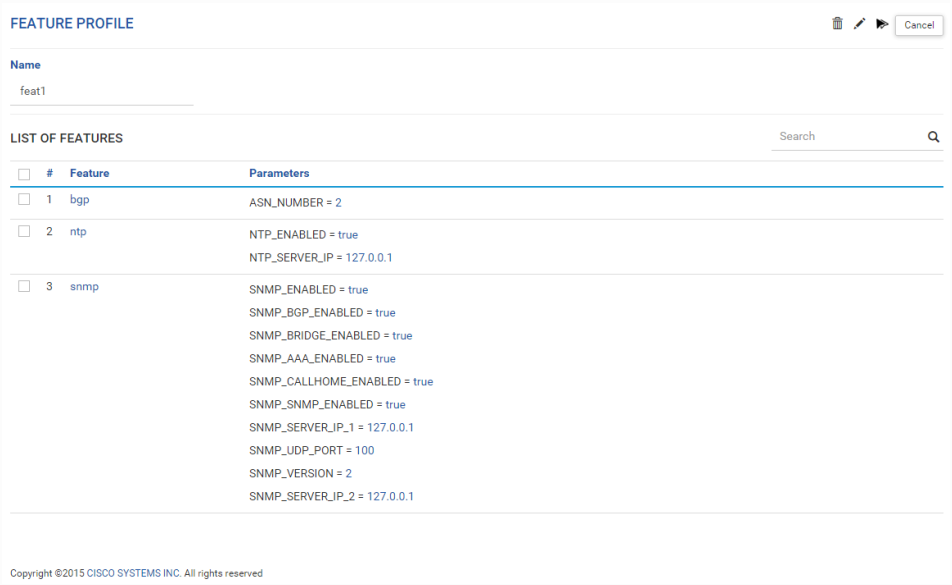


Once FEATURE is saved, it is available for use in building a FEATURE PROFILE.

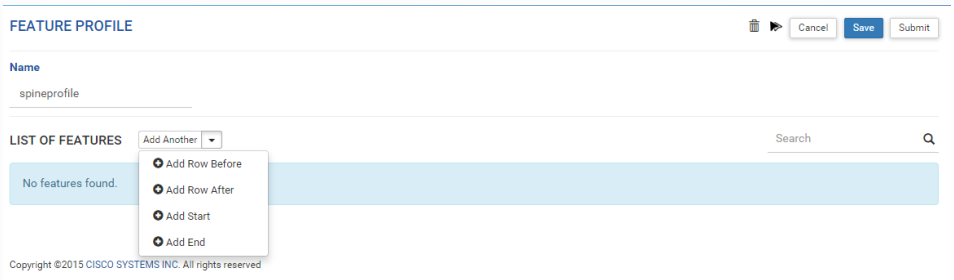
FEATURE PROFILES is a collection of FEATURES. For example, each switch in the network in addition to running BGP also requires NTP, SNMP, OSPF, VXLAN configurations. FEATURE PROFILES allows to bundle together these FEATURES and create a single profile, which then can be associated or applied to a group of similar switches.

Commented [s6]: May also require

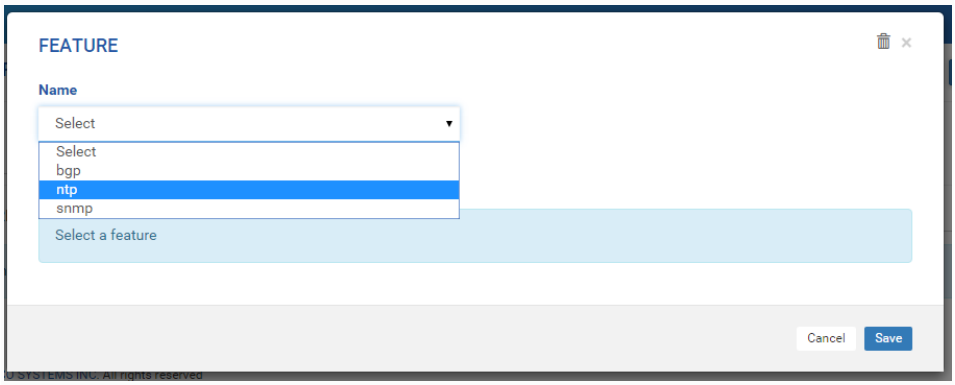
Example shows a FEATURE PROFILE named “feat1”, which groups together BGP, SNMP and NTP FEATURES.



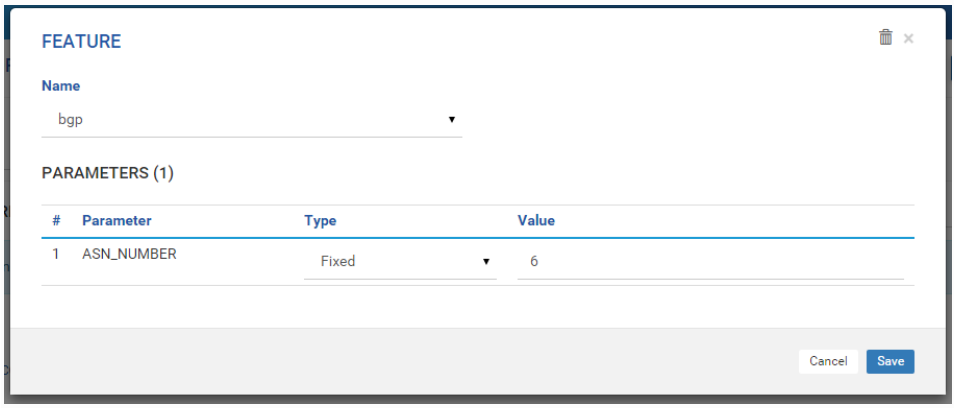
To create a new FEATURE PROFILE, click “+Add” from FEATURE PROFILES list view.



Add FEATURES to the FEATURE PROFILE using “Add Another” button. Sequence in which FEATURES get added can be selected from options available in “Add Another” button.



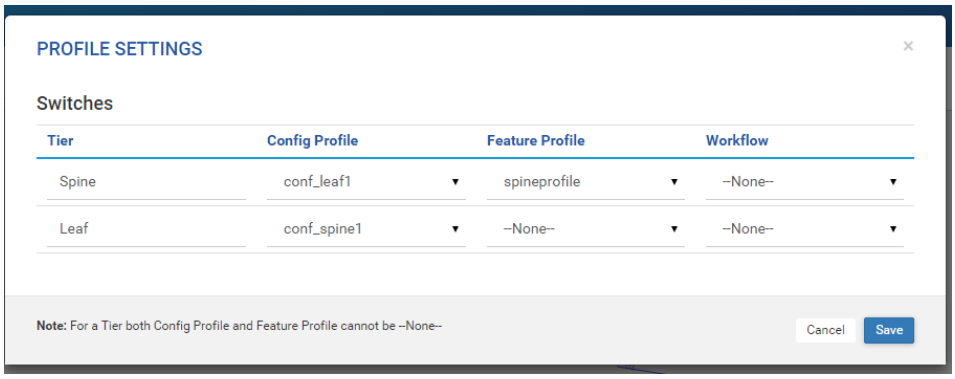
When FEATURE is added, all parameters associated with that FEATURE are displayed and parameter. Parameter substitution type and value are filled and FEATURE saved,



Once all FEATURES are added to FEATURE PROFILE, click SUBMIT to submit and make it available for use in FABRIC.

Apply FEATURE PROFILE to FABRIC

To apply FEATURE PROFILE to FABRIC, select FABRIC and in FABRIC DESIGN pane, select VIEW PROFILES and click edit. In PROFILE SETTINGS, select FEATURE PROFILE name to be assigned to each tier. Save PROFILE SETTINGS after modifying FEATURE PROFILE settings.

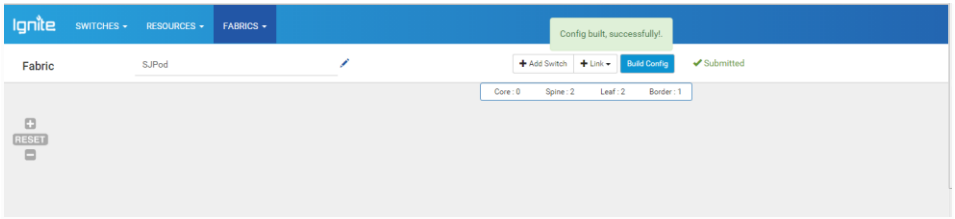


After assigning PROFILES and updating switches and links as required, SUBMIT FABRIC if not already submitted.

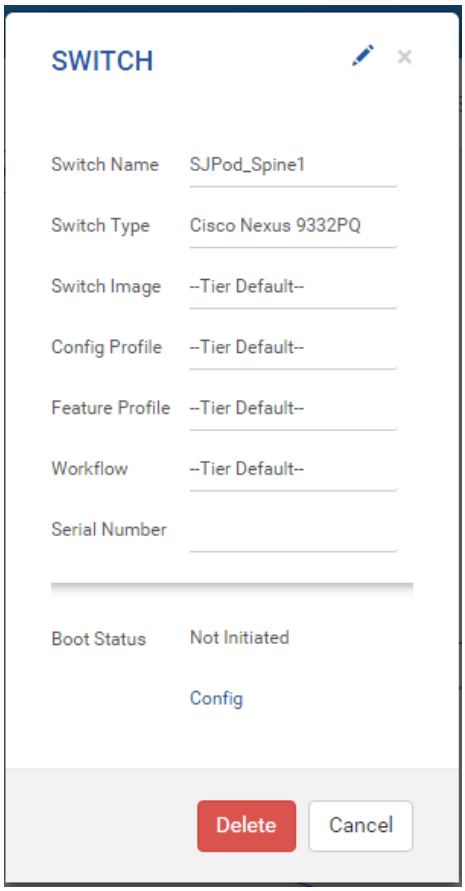
Building FETAURE PROFILE Configurations

FEATURE PROFILE Configurations are not built during POAP boot process. FEATURE PROFILEs are applied to a FABRIC or to different tiers of FABRIC. These configurations are dependent on the network topology and hence cannot be built in isolation. Depending on the size of network and FEATUREs used in FEATURE PROFILE configuration build may take time. Hence FEATURE PROFILE configurations are built in advance once FABRIC is ready for deployment. Click “BUILD CONFIG” to start configuration build. Once completed, message “Completed Successfully” is displayed.

Commented [s7]: Generally it takes few seconds.



To view configuration generated by FEATURE PROFILE settings, select a switch from FABRIC DESIGN view and click to view switch details. Click on “Config” link to open the configuration generated.



Example configuration generated by FEATURE PROFILE.

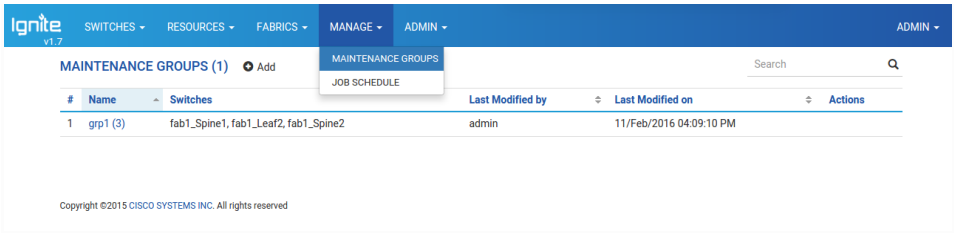
Manage

It can be used to schedule jobs of multiple tasks. Each task can be applied on a group of switches. Supported tasks are upgrading the system image or epld image on a group of switches.

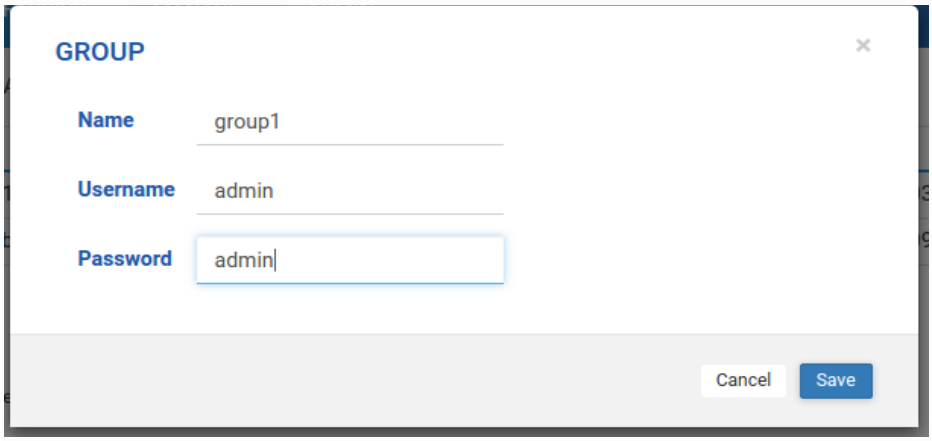
Adding a new GROUP

To edit, view or add Groups select “MAINTENANCE GROUPS option from “MANAGE” tab in top menu. List of image profiles is displayed.

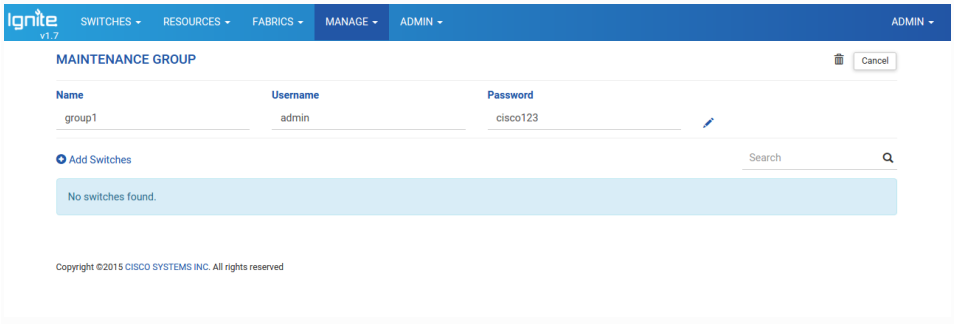
1. To view GROUP, select “MAINTENANCE GROUPS” option in “MANAGE” tab. List view of available GROUPS is displayed.



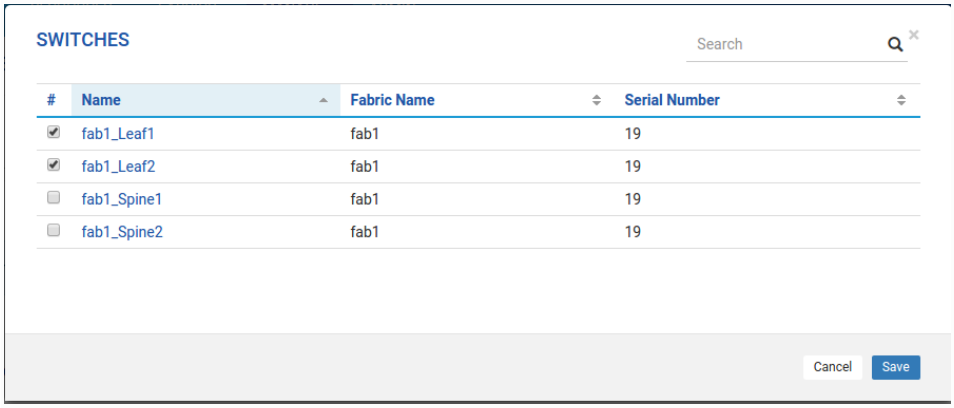
2. Add new GROUP by clicking “+Add”, fill out details.



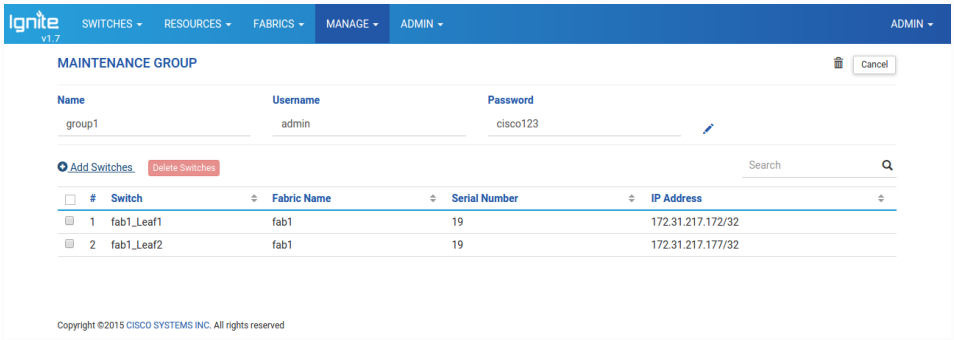
3. Add Booted Switches by clicking “+Add”.



4. After selecting the switches click “Save” to save the switches



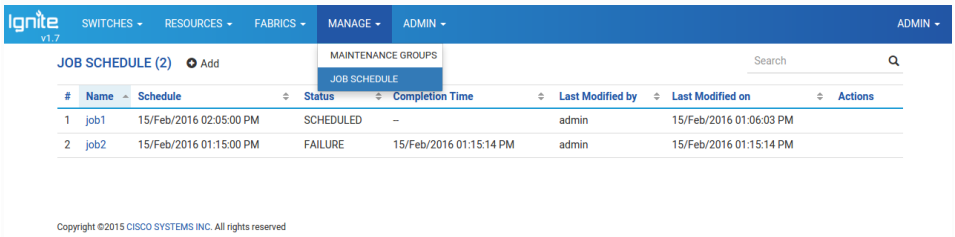
5. Once submitted, GROUP will be available to be applied in JOB SCHEDULE



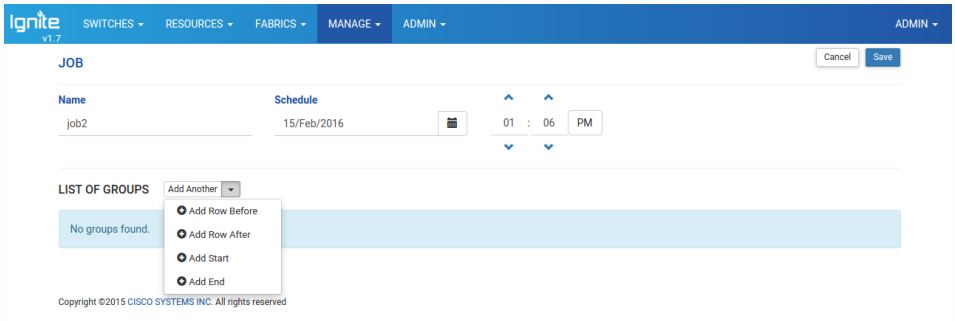
JOB SCHEDULE

Adding a new JOB

1. To add a new JOB, select “JOB SCHEDULE” option in “MANAGE” tab. List view of available JOBS is displayed.



2. Add new JOB by clicking “+Add”, fill out details



Assign Name to JOB and Schedule Time to run this particular period, define the list of GROUPs used to build this task by selecting “Add Another” button and fill out details

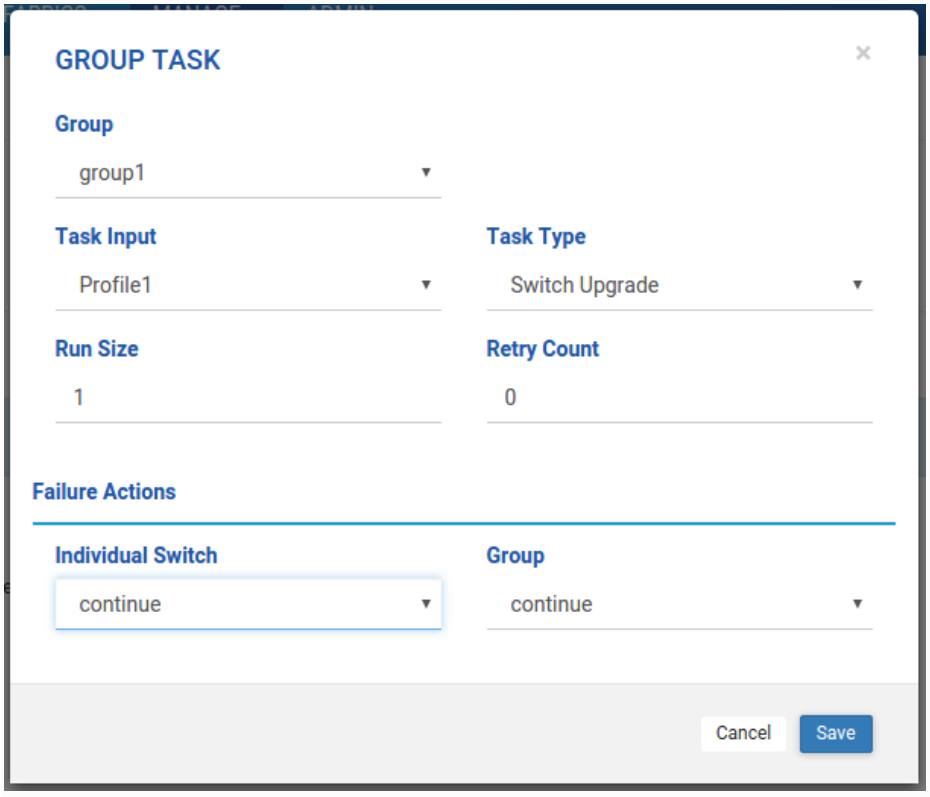
You can add, delete Groups in JOB during edit process.

Following operations are allowed:

Add Row Before – allows adding a GROUP TASK before a selected row in the GROUPS list
Add Row After – allows adding a GROUP TASK after a selected row in the GROUPS list
Add Start – allows adding a GROUP TASK at the start of GROUPS list
Add End – allows adding a GROUP TASK at the end of GROUPS list

A row is selected for operation, by clicking on the selection box.

3. To add a new GROUP TASK, click “+Add” button from GROUPS list view and fill out details.



Following fields in GROUP TASK form should be filled and saved.

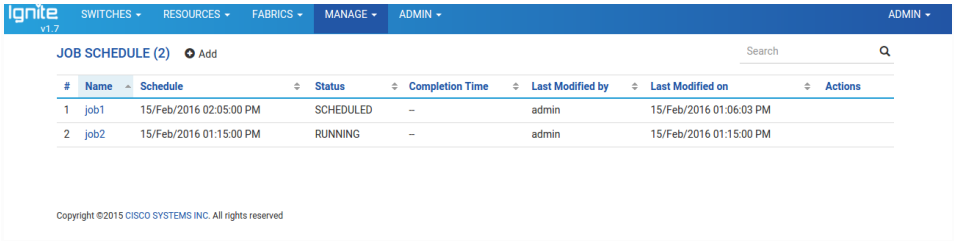
Filed	Description
Group	Name of Group used to build the Task . Values can be filled from the drop down available in this option. Only Groups submitted, will be available for selection.
Task Input	Name of Image Profile used to build on each switch . Values can be filled from the drop down available in this option. Only Image Profile submitted, will be available for selection.
Task Type	Type of the Task used to build on the switches. Values can be filled from the drop down available in this option. Only Switch Upgrade and Epld Upgrade submitted, will be available for selection. ***Note: EPLD upgrade is not supported***
Run Size	A number used in Parallel or Sequential Execution of task .
Retry Count	A number used to run the task for certain times if any failure occurs .

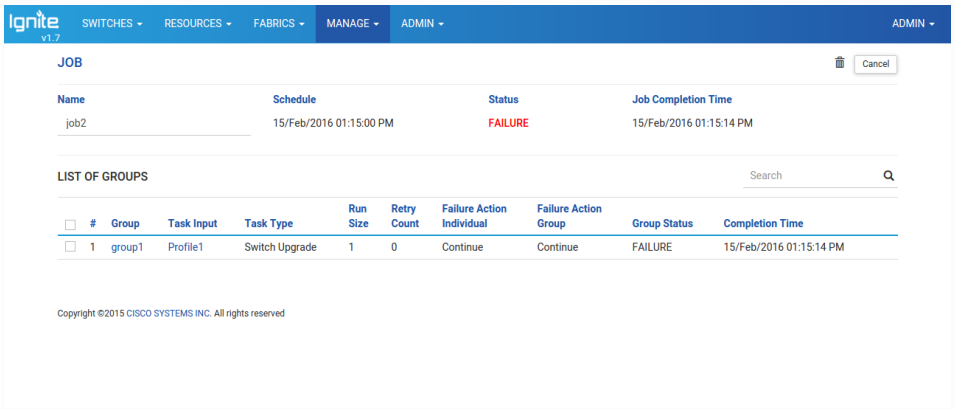
Following fields in Failure Actions form should be filled and saved.

Filed	Description
Individual Switch	Action used to run on each switch failure. Values can be filled from the drop down available in this option. Only continue and abort option, will be available for selection.
Group	Action used to run on Group failure. Values can be filled from the drop down available in this option. Only continue and abort option, will be available for selection

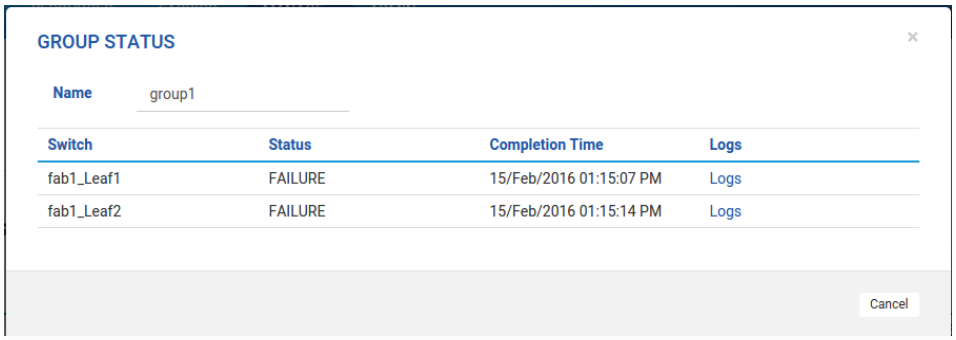
The following are the status of JOB

1. SCHEDULED : Job submitted before the Scheduled period
2. RUNNING : After the expiration of Schedule period the JOB will be in Execution which is known as RUNNING
3. FAILURE : After the completion of execution of JOB if any error occurs then the JOB status will be in FAILURE
4. SUCCESS : If there is no error in execution of JOB then status will be SUCCESS





4.To see the error message click on the logs



Error message for the FAILURE of JOB