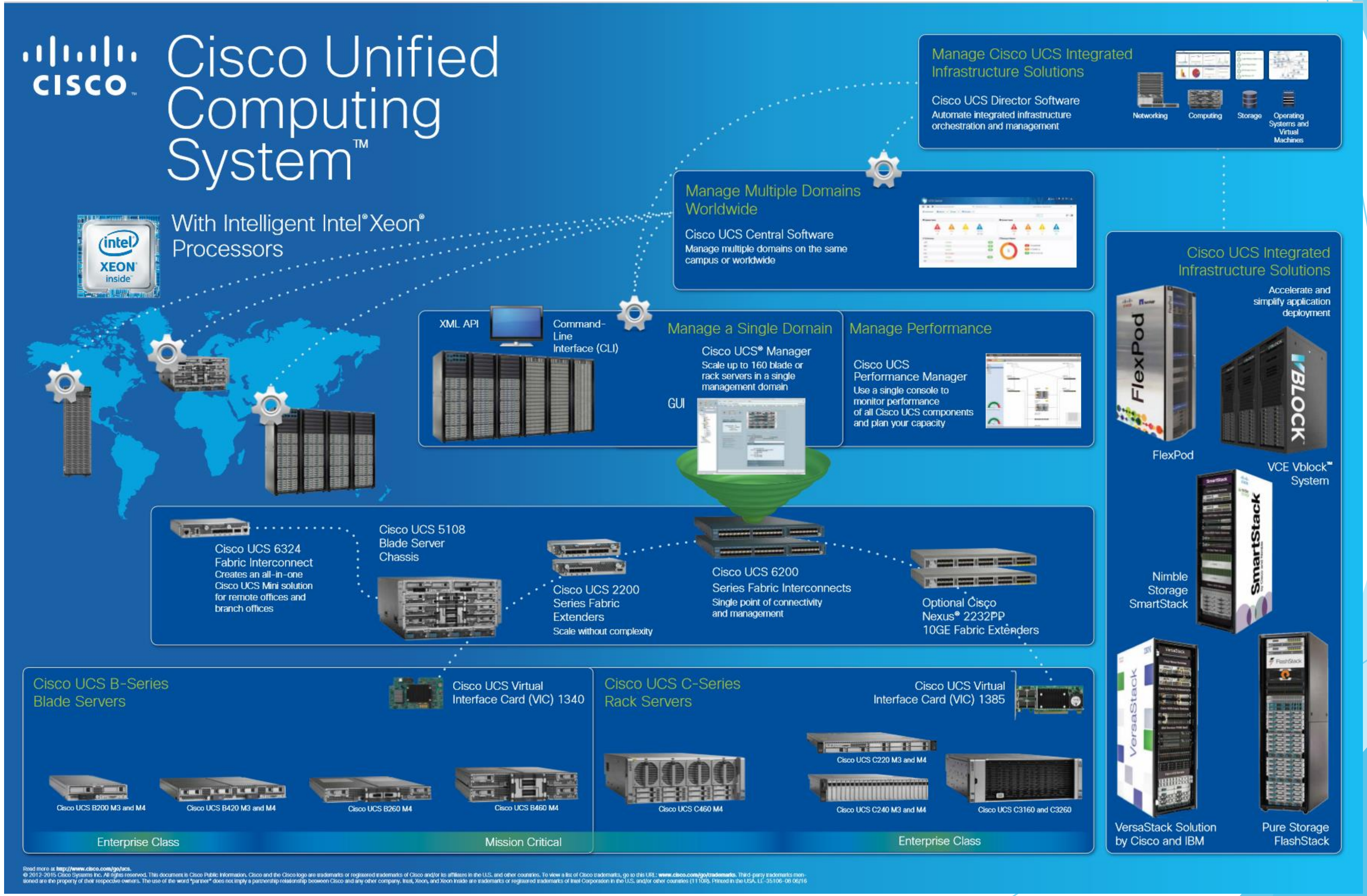
Ansible-Python SDK FOR Cisco UCS Manager Documentation

By:

Jyotsna Venkatesh

Cisco UCS Product Family



Cisco Unified Computing System™

With Intelligent Intel® Xeon® Processors

Manage Cisco UCS Integrated Infrastructure Solutions

Cisco UCS Director Software
Automate integrated infrastructure orchestration and management

Manage Multiple Domains Worldwide

Cisco UCS Central Software
Manage multiple domains on the same campus or worldwide

Manage a Single Domain

Cisco UCS® Manager
Scale up to 160 blade or rack servers in a single management domain

Manage Performance

Cisco UCS Performance Manager
Use a single console to monitor performance of all Cisco UCS components and plan your capacity

Cisco UCS 6324 Fabric Interconnect
Creates an all-in-one Cisco UCS Mini solution for remote offices and branch offices

Cisco UCS 5108 Blade Server Chassis

Cisco UCS 2200 Series Fabric Extenders
Scale without complexity

Cisco UCS 6200 Series Fabric Interconnects
Single point of connectivity and management

Optional Cisco Nexus® 2232PP 10GE Fabric Extenders

Cisco UCS B-Series Blade Servers

Cisco UCS Virtual Interface Card (VIC) 1340

Cisco UCS C-Series Rack Servers

Cisco UCS Virtual Interface Card (VIC) 1385

Cisco UCS Integrated Infrastructure Solutions

Accelerate and simplify application deployment

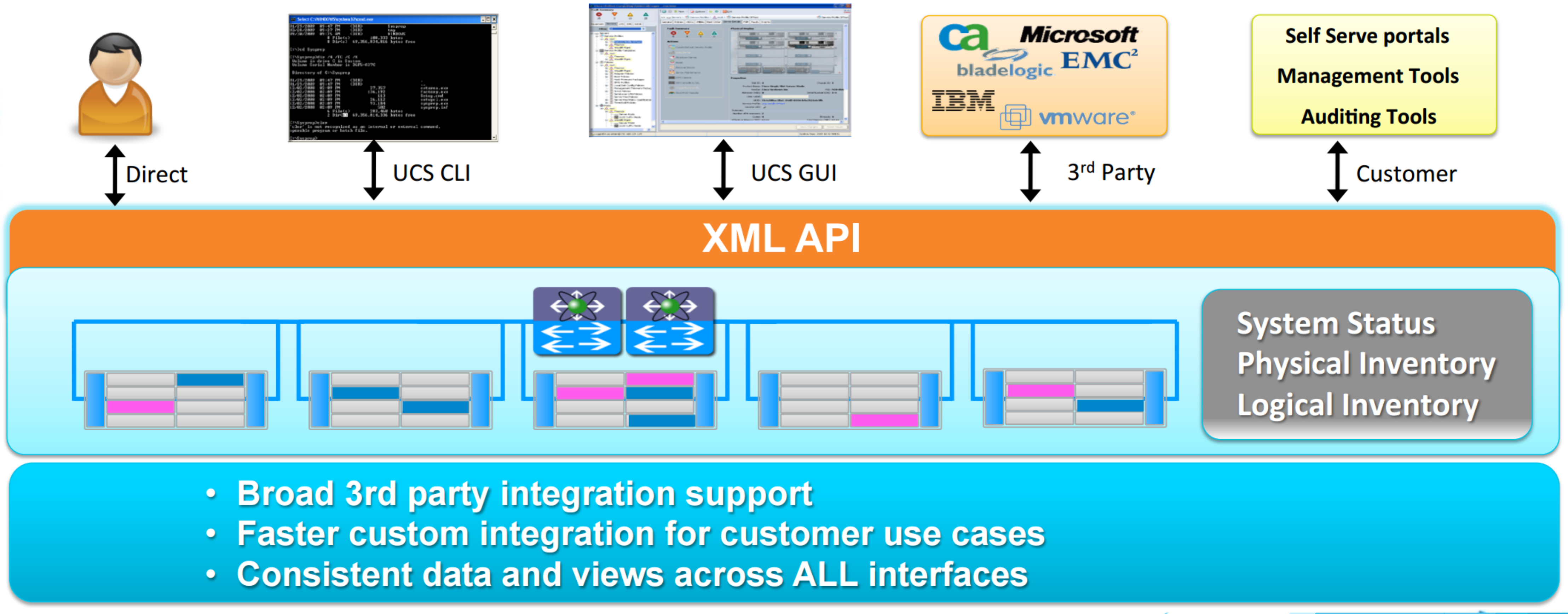
FlexPod
VCE Vblock™ System
Nimble Storage SmartStack
VersaStack Solution by Cisco and IBM
Pure Storage FlashStack

Attractive features of UCSM:

- Automate routine tasks to increase agility, simplify daily operations, and to reduce management and administration expenses.
- Supports Cisco UCS B-Series Blade and C-Series Rack Servers, the C3260 storage server, Cisco UCS Mini, and the Cisco HyperFlex hyperconverged infrastructure
- Programmatically controls server, network, and storage resources
- Policy-driven management, so they can be efficiently managed at scale through software
- Works with HTML 5, Java, or CLI graphical user interfaces
- Can automatically detect, inventory, manage, and provision system components that are added or changed
- Facilitates integration with third-party systems management tools
- Builds on existing skills and supports collaboration across disciplines through role-based administration

XML API Infrastructure

- Bi-Directional access to physical & logical internals



- Broad 3rd party integration support
- Faster custom integration for customer use cases
- Consistent data and views across ALL interfaces

Managed Information Model

- All the physical and logical components that comprise Cisco UCS are represented in a hierarchical Management Information Model, referred to as the Management Information Tree (MIT). Each node in the tree represents a Managed Object (MO), uniquely identified by its Distinguished Name. (DN)

```
Tree (topRoot)              Distinguished Name
|– sys                      sys
|– chassis-1                    sys/chassis-1
|– chassis-2                    sys/chassis-2
|– chassis-3                    sys/chassis-3
    |– blade-1              sys/chassis-3/blade-1
    |       |– adaptor-1        sys/chassis-3/blade-1/adaptor-1
    |– blade-2              sys/chassis-3/blade-2
            |– adaptor-1        sys/chassis-3/blade-2/adaptor-1
            |– adaptor-2        sys/chassis-3/blade-2/adaptor-2
```

Managed objects

- What is a managed object?
- What is a distinguished name?
- What is a relative name?

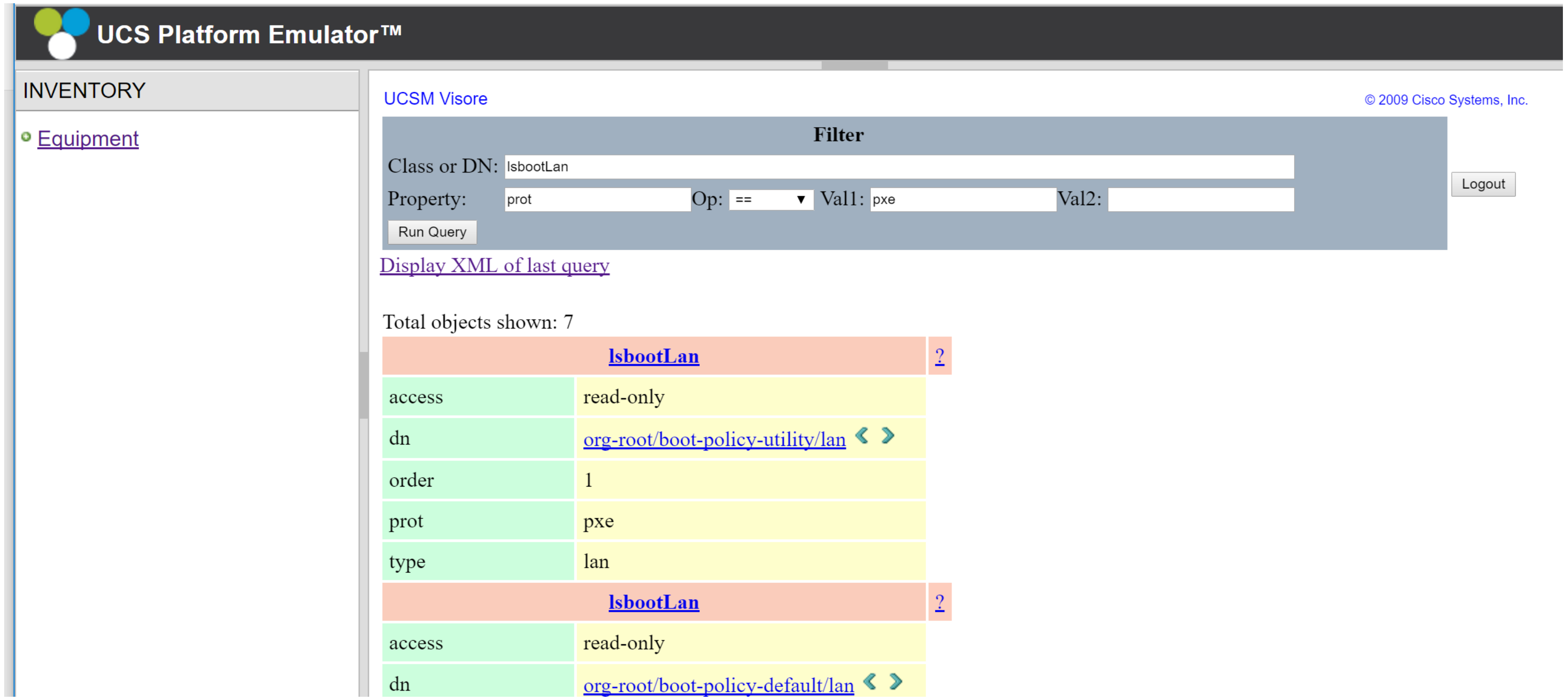
Every Managed Object is uniquely identified in the tree with its Distinguished Name (Dn) and can be uniquely identified within the context of its parent with its Relative Name (Rn).

The Dn identifies the place of the MO in the MIT. A Dn is a concatenation of all the relative names starting from the root to the MO itself.

Essentially, Dn = [Rn]/[Rn]/[Rn]/…/[Rn]

```
<dn = “sys/chassis-5/blade-2/adaptor-1” />
```

Use case example of XML API - Visore


UCS Platform Emulator™

INVENTORY

Equipment

UCSM Visore

© 2009 Cisco Systems, Inc.

Filter

Class or DN: lsbootLan

Property: prot Op: == Val1: pxe Val2:

Run Query

Logout

Display XML of last query

Total objects shown: 7

lsbootLan	?
access	read-only
dn	org-root/boot-policy-utility/lan
order	1
prot	pxe
type	lan
lsbootLan	?
access	read-only
dn	org-root/boot-policy-default/lan

Python SDK for UCSM and demo

- Login and launch Java GUI using console

```
from ucsmsdk.utils.ucsguilaunch import ucs_gui_launch
from ucsmsdk.ucshandle import UcsHandle
# Login to the server
handle = UcsHandle(<ip>, <username>, <password>)
handle.login() # launch the UCSM GUI
ucs_gui_launch(handle)
```

Convert to python tool demo

- from ucsmsdk.utils.converttopython import convert_to_ucs_python
- convert_to_ucs_python()

```python
from ucsmsdk.utils.converttopython import convert_to_ucs_python
convert_to_ucs_python(dump_xml=True)
```

Querying the Managed Information Model

- Example query:

```
a = handle.query_children(in_dn="org-root", class_id="LsbootPolicy",
filter_str='(name,"sample3", type="eq") and (descr,"sample3", type="eq")')
```

```
handle.query_dn("org-root/boot-policy-sample3")
```

```
handle.query_children(in_dn="org-root",
class_id="LsbootPolicy")
```

Ansible Introduction

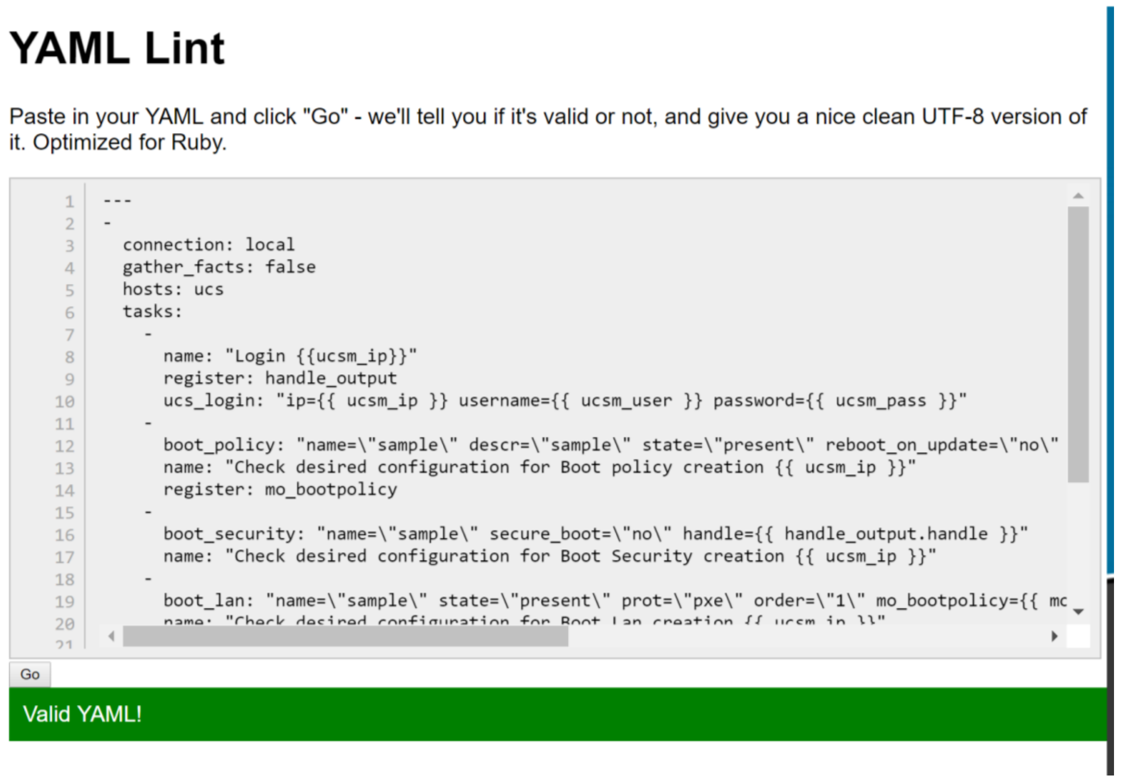
- Configuration management and orchestration
- What is a playbook?
- Each **playbook** contains one or more plays, which map hosts to a certain function.
- **Ansible** does this through something called tasks, which are basically module calls.

```
---
- hosts: droplets
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

  handlers:
    - name: start nginx
      service: name=nginx state=started
```

Basics of Ansible syntax and writing/compiling a playbook

- Demo of YamlLint

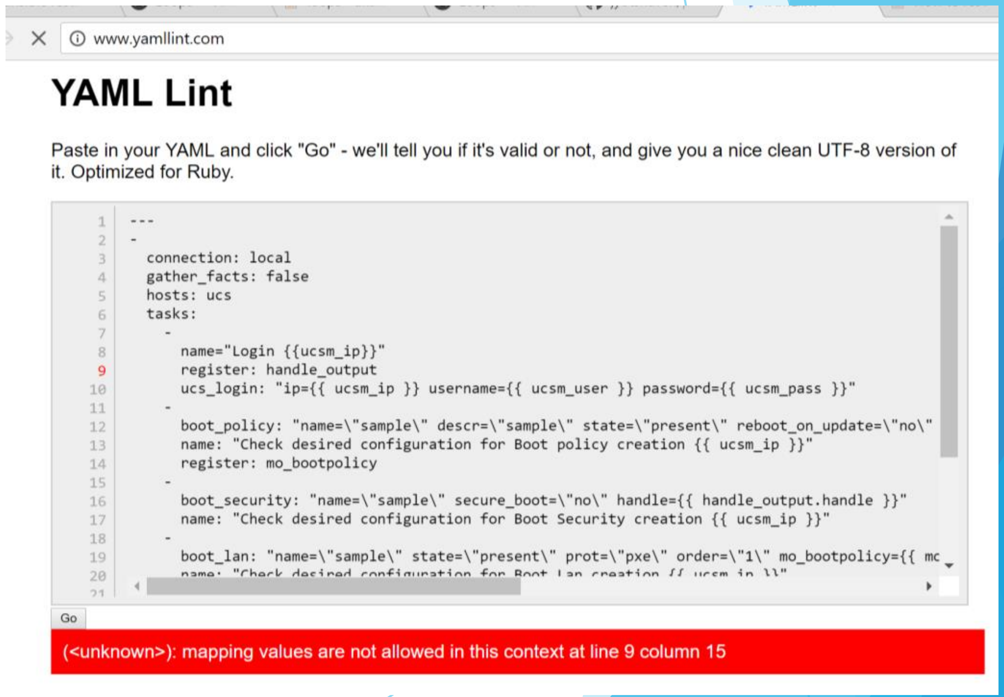


YAML Lint

Paste in your YAML and click "Go" - we'll tell you if it's valid or not, and give you a nice clean UTF-8 version of it. Optimized for Ruby.

```
 1  ---
 2  -
 3    connection: local
 4    gather_facts: false
 5    hosts: ucs
 6    tasks:
 7      -
 8        name: "Login {{ucsm_ip}}"
 9        register: handle_output
10        ucs_login: "ip={{ ucsm_ip }} username={{ ucsm_user }} password={{ ucsm_pass }}"
11      -
12        boot_policy: "name=\"sample\" descr=\"sample\" state=\"present\" reboot_on_update=\"no\"
13        name: "Check desired configuration for Boot policy creation {{ ucsm_ip }}"
14        register: mo_bootpolicy
15      -
16        boot_security: "name=\"sample\" secure_boot=\"no\" handle={{ handle_output.handle }}"
17        name: "Check desired configuration for Boot Security creation {{ ucsm_ip }}"
18      -
19        boot_lan: "name=\"sample\" state=\"present\" prot=\"pxe\" order=\"1\" mo_bootpolicy={{ mc
20        name: "Check desired configuration for Boot Lan creation {{ ucsm_ip }}"
21
```

Valid YAML!



YAML Lint

Paste in your YAML and click "Go" - we'll tell you if it's valid or not, and give you a nice clean UTF-8 version of it. Optimized for Ruby.

```
 1  ---
 2  -
 3    connection: local
 4    gather_facts: false
 5    hosts: ucs
 6    tasks:
 7      -
 8        name="Login {{ucsm_ip}}"
 9        register: handle_output
10        ucs_login: "ip={{ ucsm_ip }} username={{ ucsm_user }} password={{ ucsm_pass }}"
11      -
12        boot_policy: "name=\"sample\" descr=\"sample\" state=\"present\" reboot_on_update=\"no\"
13        name: "Check desired configuration for Boot policy creation {{ ucsm_ip }}"
14        register: mo_bootpolicy
15      -
16        boot_security: "name=\"sample\" secure_boot=\"no\" handle={{ handle_output.handle }}"
17        name: "Check desired configuration for Boot Security creation {{ ucsm_ip }}"
18      -
19        boot_lan: "name=\"sample\" state=\"present\" prot=\"pxe\" order=\"1\" mo_bootpolicy={{ mc
20        name: "Check desired configuration for Boot Lan creation {{ ucsm_ip }}"
21
```

```
(<unknown>): mapping values are not allowed in this context at line 9 column 15
```

Development set-up

- Step 1: Linux environment/ Vmware Workstation Pro 12
- Step 2: Ansible
  - wget http://releases.ansible.com/ansible/ansible-2.0.0-0.9.rc4.tar.gz
  - tar -xvzf ansible-2.0.0-0.9.rc4.tar.gz
  - cd ansible-2.0.0
  - sudo python setup.py install
- Step 3: Python 2.7, pip (http://www.howtogeek.com/197947/how-to-install-python-on-windows/)
- Step 4: UCSPE -https://communities.cisco.com/docs/DOC-37827
- Step 5: git clone ucsmsdk
- Step 6: git clone https://github.com/jyotsnaven/python-ansible-ucsm

Login module

- module: ucs_login
- Short_description: Login
- Description:
-   - Allows user to login
- Example API: handle.login()

Boot policy module

- module: boot_policy
- Short_description: Create, modify or remove boot policy
- Description:
- - Allows to check if boot policy exists. If present, check for desired configuration. If desired config is not present, apply settings. If boot policy is not present, create and apply desired settings. If the desired state is 'absent', remove boot policy if it is currently present
- Example API: mo = LsbootPolicy(parent_mo_or_dn="org-root", name="newdemo22", descr="newdemo", reboot_on_update="no", policy_owner="local", enforce_vnic_name="yes", boot_mode="legacy")

Boot lan module

- module: boot_lan
- Short description: Create, modify or remove boot lan
- Description:
- - Allows to check if boot lan mo exists. If present, check for desired configuration. If desired config is not present, apply settings. If boot lan mo is not present, create and apply desired settings. If the desired state is 'absent', remove boot lan mo if it is currently present. If the desired state is 'absent', remove boot policy if it is currently present
- Example API: mo = LsbootLan(parent_mo_or_dn=obj, prot="pxe", order="3")

Boot security module

- module: boot_security
- Short description: Apply desired boot security settings for boot policy
- description:
- - Allows to check if desired boot security option is selected. If not modify settings to apply the desired secure_boot option for boot policy.
- Example API: mo_1 = LsbootBootSecurity(parent_mo_or_dn=mo, secure_boot="yes")

Boot lan vnic module

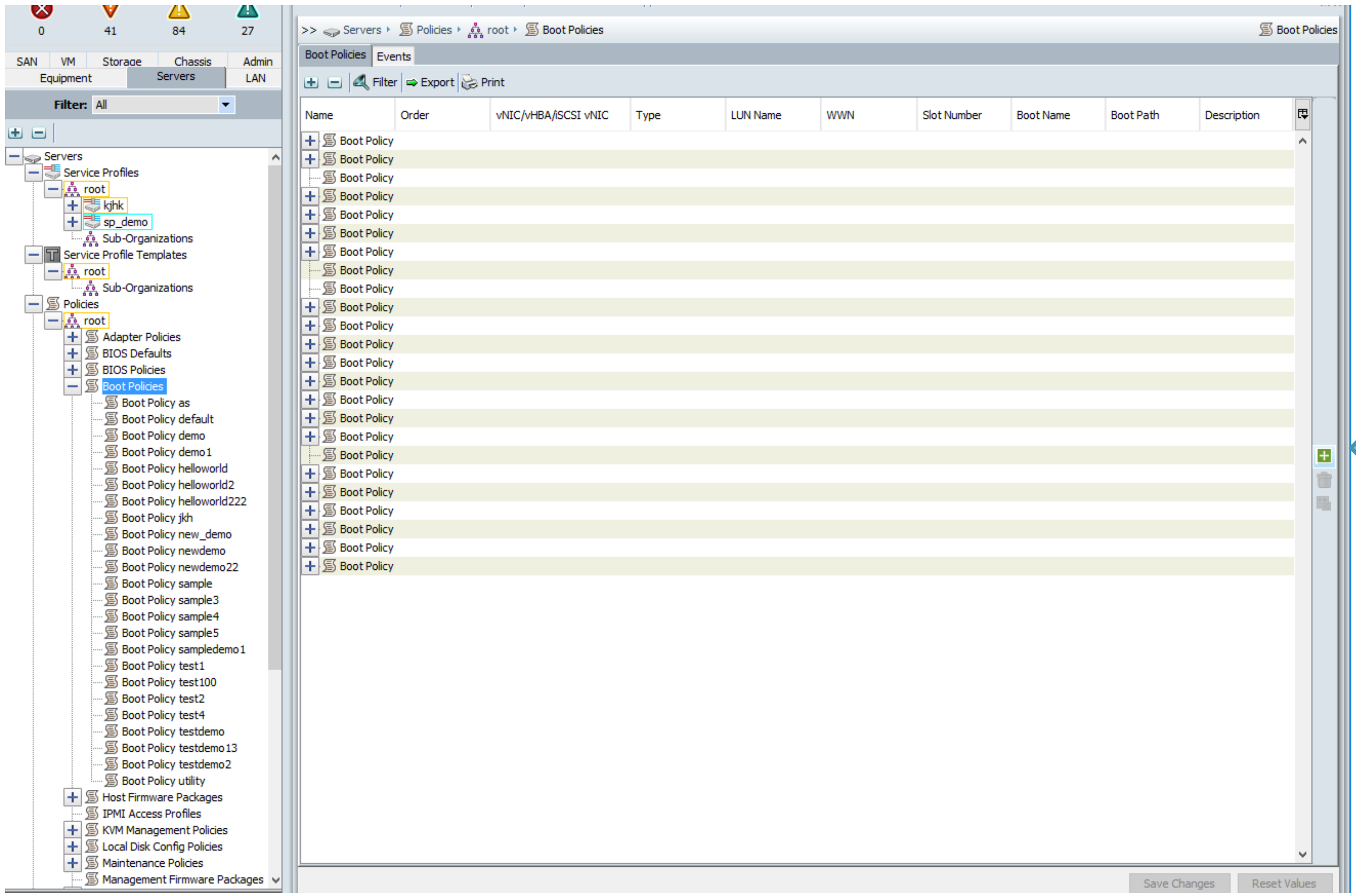
- module: boot_lan_vnic
- Short description: Create, modify or remove boot lan vnic primary and secondary
- description:
- Allows to check if boot lan vnic mo exists. If present, add a secondary boot lan vnic mo. If boot lan vnic mo is not present, create a primary boot lan vnic mo. If the desired state is 'absent', remove boot lan mo if it is currently present. If it is secondary, do nothing. If it is primary, make current secondary vnic primary and remove moExample API: mo = LsbootPolicy(parent_mo_or_dn="org-root", name="newdemo22", descr="newdemo", reboot_on_update="no", policy_owner="local", enforce_vnic_name="yes", boot_mode="legacy")
- Example API: mo_1 = LsbootLanImagePath(parent_mo_or_dn=mo, prov_srv_policy_name="", img_sec_policy_name="", vnic_name="pri", i_scsi_vnic_name="", boot_ip_policy_name="", img_policy_name="", type="primary")

Logout module

- module: ucs_logout
- Short_description: Logout
- Description:
- - Allows user to logout
- Example API: handle.logout()

Create boot policy UI

- Step 1:



0 41 84 27

SAN VM Storage Chassis Admin
Equipment Servers LAN

Filter: All

```
Servers
  Service Profiles
    root
      kjhk
      sp_demo
      Sub-Organizations
  Service Profile Templates
    root
      Sub-Organizations
  Policies
    root
      Adapter Policies
      BIOS Defaults
      BIOS Policies
      Boot Policies
        Boot Policy as
        Boot Policy default
        Boot Policy demo
        Boot Policy demo1
        Boot Policy helloworld
        Boot Policy helloworld2
        Boot Policy helloworld222
        Boot Policy jkh
        Boot Policy new_demo
        Boot Policy newdemo
        Boot Policy newdemo22
        Boot Policy sample
        Boot Policy sample3
        Boot Policy sample4
        Boot Policy sample5
        Boot Policy sampledemo1
        Boot Policy test1
        Boot Policy test100
        Boot Policy test2
        Boot Policy test4
        Boot Policy testdemo
        Boot Policy testdemo13
        Boot Policy testdemo2
        Boot Policy utility
      Host Firmware Packages
      IPMI Access Profiles
      KVM Management Policies
      Local Disk Config Policies
      Maintenance Policies
      Management Firmware Packages
```

>> Servers › Policies › root › Boot Policies — Boot Policies

Boot Policies | Events

Filter | Export | Print

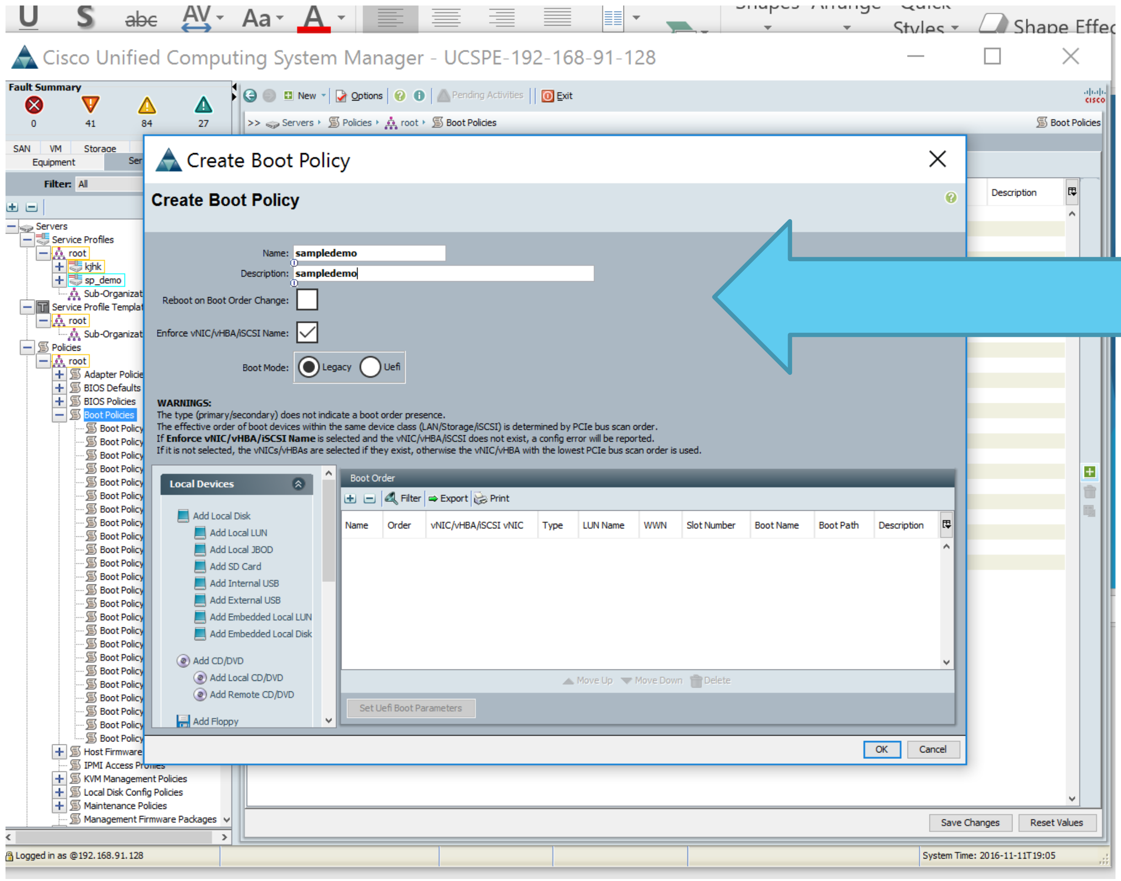
Name	Order	vNIC/vHBA/iSCSI vNIC	Type	LUN Name	WWN	Slot Number	Boot Name	Boot Path	Description
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									
Boot Policy									

Save Changes | Reset Values

Navigate to Servers tab. Click on this icon

Boot policy creation UI

- Step 2:



Create Boot Policy

Name: sampledemo
Description: sampledemo
Reboot on Boot Order Change: ☐
Enforce vNIC/vHBA/iSCSI Name: ✓
Boot Mode: [✓ Legacy] Uefi

WARNINGS:
The type (primary/secondary) does not indicate a boot order presence.
The effective order of boot devices within the same device class (LAN/Storage/iSCSI) is determined by PCIe bus scan order.
If **Enforce vNIC/vHBA/iSCSI Name** is selected and the vNIC/vHBA/iSCSI does not exist, a config error will be reported.
If it is not selected, the vNICs/vHBAs are selected if they exist, otherwise the vNIC/vHBA with the lowest PCIe bus scan order is used.

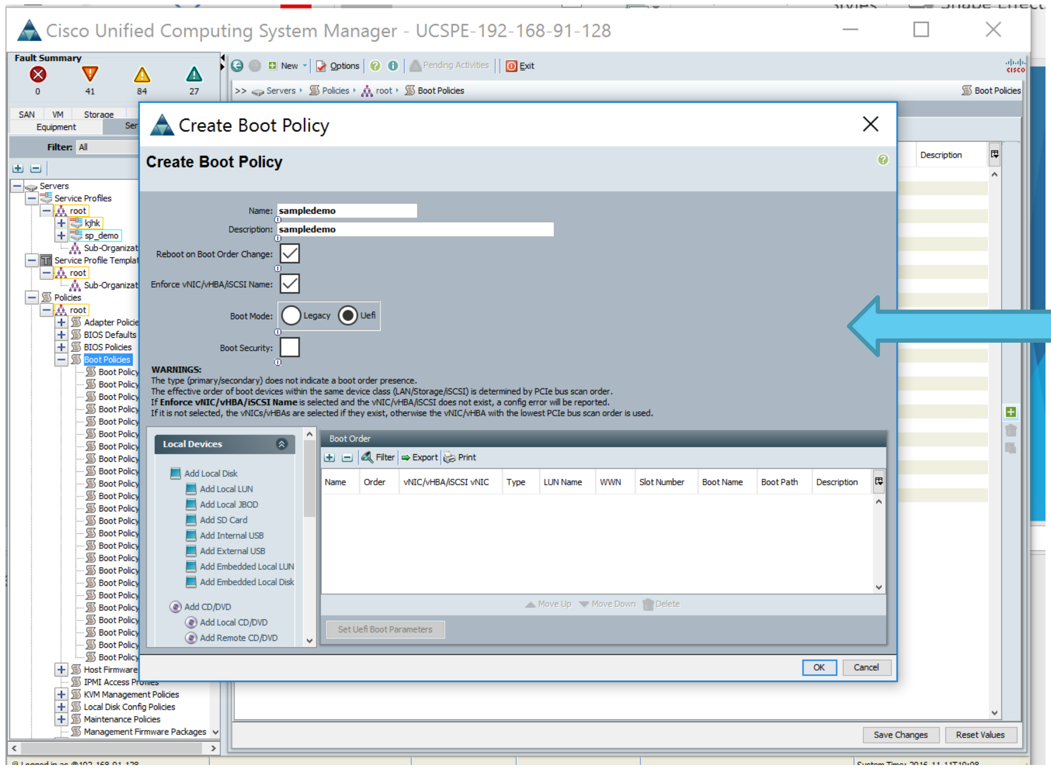
Name	Order	vNIC/vHBA/iSCSI vNIC	Type	LUN Name	WWN	Slot Number	Boot Name	Boot Path	Description

Logged in as @192.168.91.128
System Time: 2016-11-11T19:05

Enter values for Name, description, reboot_on_update,policy_owner and enforce_vnic_name

Boot policy creation UI

- Step 3:



Cisco Unified Computing System Manager - UCSPE-192-168-91-128
Fault Summary
Servers › Policies › root › Boot Policies
Create Boot Policy

Create Boot Policy

Name: sampledemo

Description: sampledemo

Reboot on Boot Order Change: ✓

Enforce vNIC/vHBA/iSCSI Name: ✓

Boot Mode: Legacy [✓ Uefi]

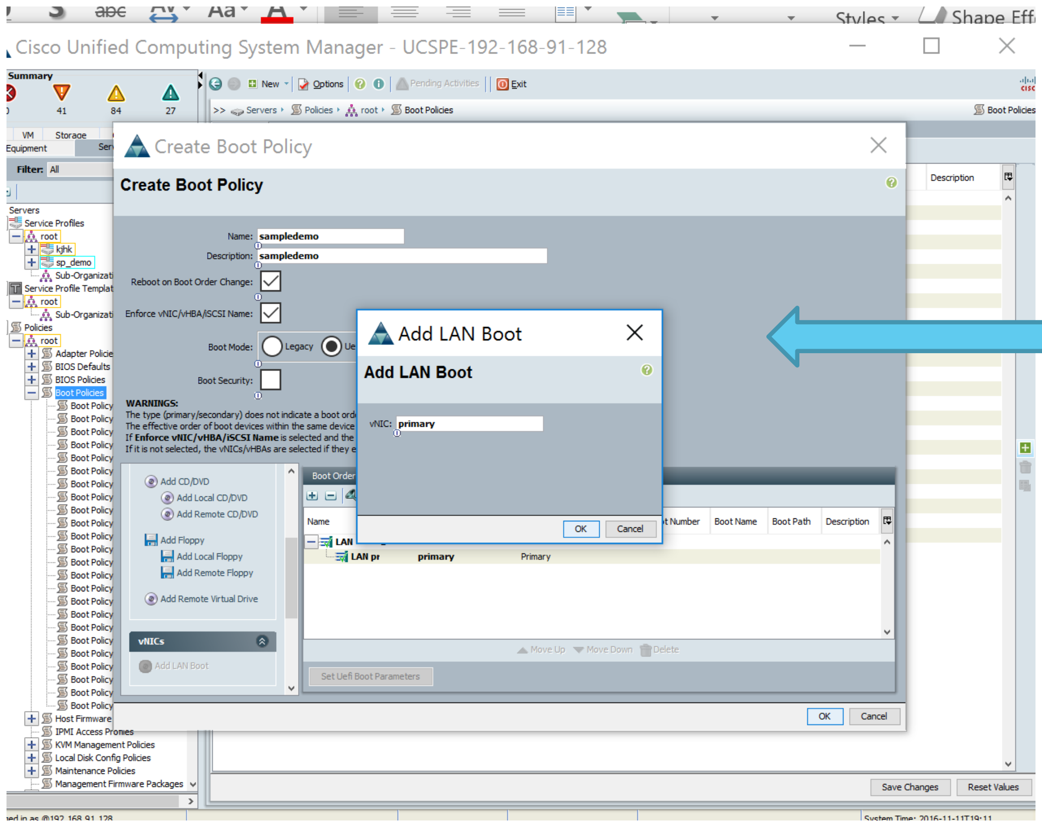
Boot Security: ☐

WARNINGS:
The type (primary/secondary) does not indicate a boot order presence.
The effective order of boot devices within the same device class (LAN/Storage/iSCSI) is determined by PCIe bus scan order.
If **Enforce vNIC/vHBA/iSCSI Name** is selected and the vNIC/vHBA/iSCSI does not exist, a config error will be reported.
If it is not selected, the vNICs/vHBAs are selected if they exist, otherwise the vNIC/vHBA with the lowest PCIe bus scan order is used.

Boot security becomes visible only for boot_mode option "uefi"

Boot policy creation UI

- Step 4:



Cisco Unified Computing System Manager - UCSPE-192-168-91-128

Create Boot Policy

Name: sampledemo
Description: sampledemo
Reboot on Boot Order Change: ✓
Enforce vNIC/vHBA/iSCSI Name: ✓
Boot Mode: Legacy [✓ Ue]
Boot Security: ☐

WARNINGS:
The type (primary/secondary) does not indicate a boot ord
The effective order of boot devices within the same device
If **Enforce vNIC/vHBA/iSCSI Name** is selected and the
If it is not selected, the vNICs/vHBAs are selected if they e

Add LAN Boot

vNIC: primary

Name			t Number	Boot Name	Boot Path	Description
LAN						
LAN pr	primary	Primary				

Add boot_lan_vnic to boot order.

- Step 5: Save Changes and click OK!