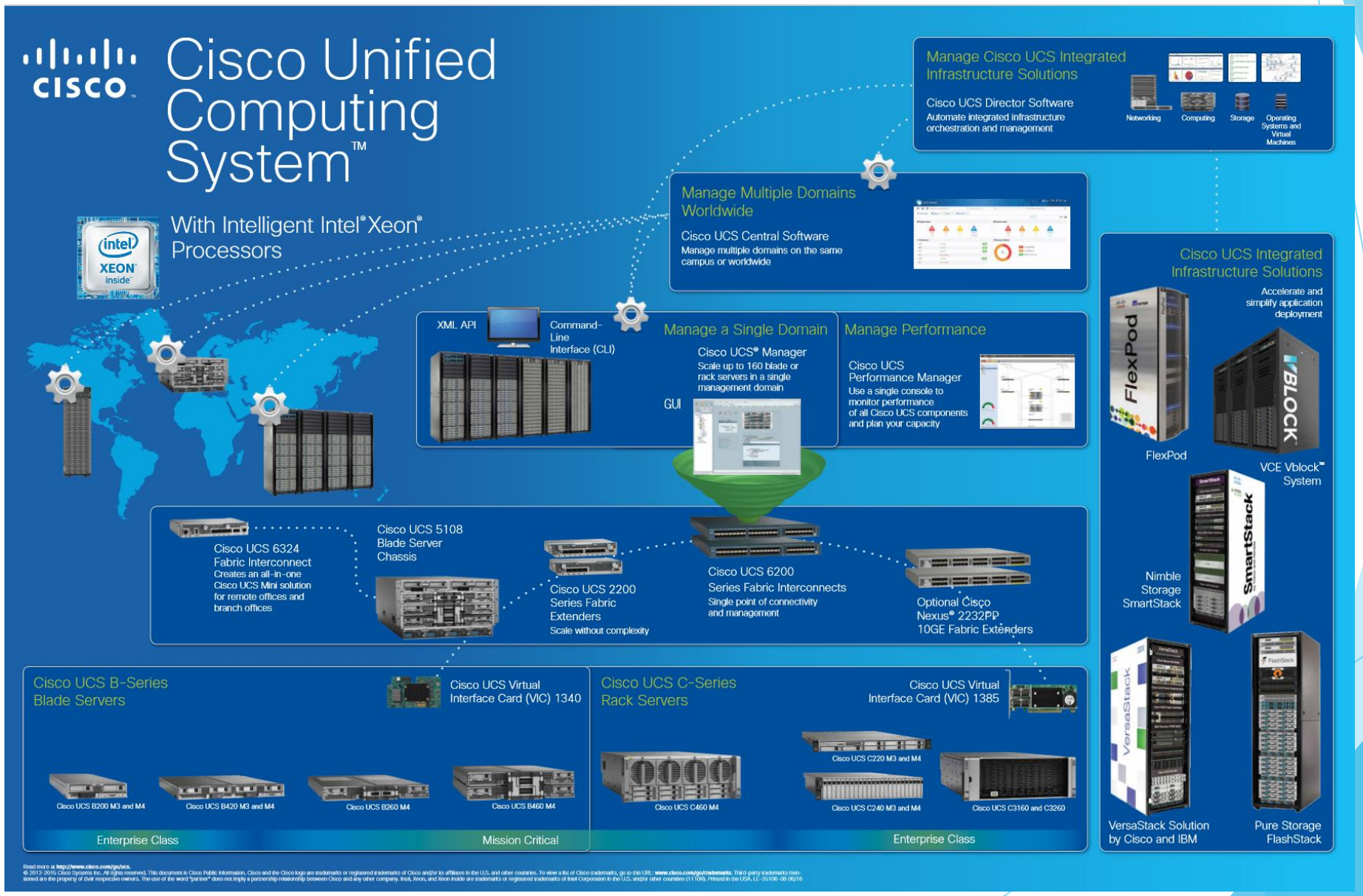
Ansible-Python SDK FOR Cisco UCS Manager Documentation

By:

Jyotsna Venkatesh

Cisco UCS Product Family

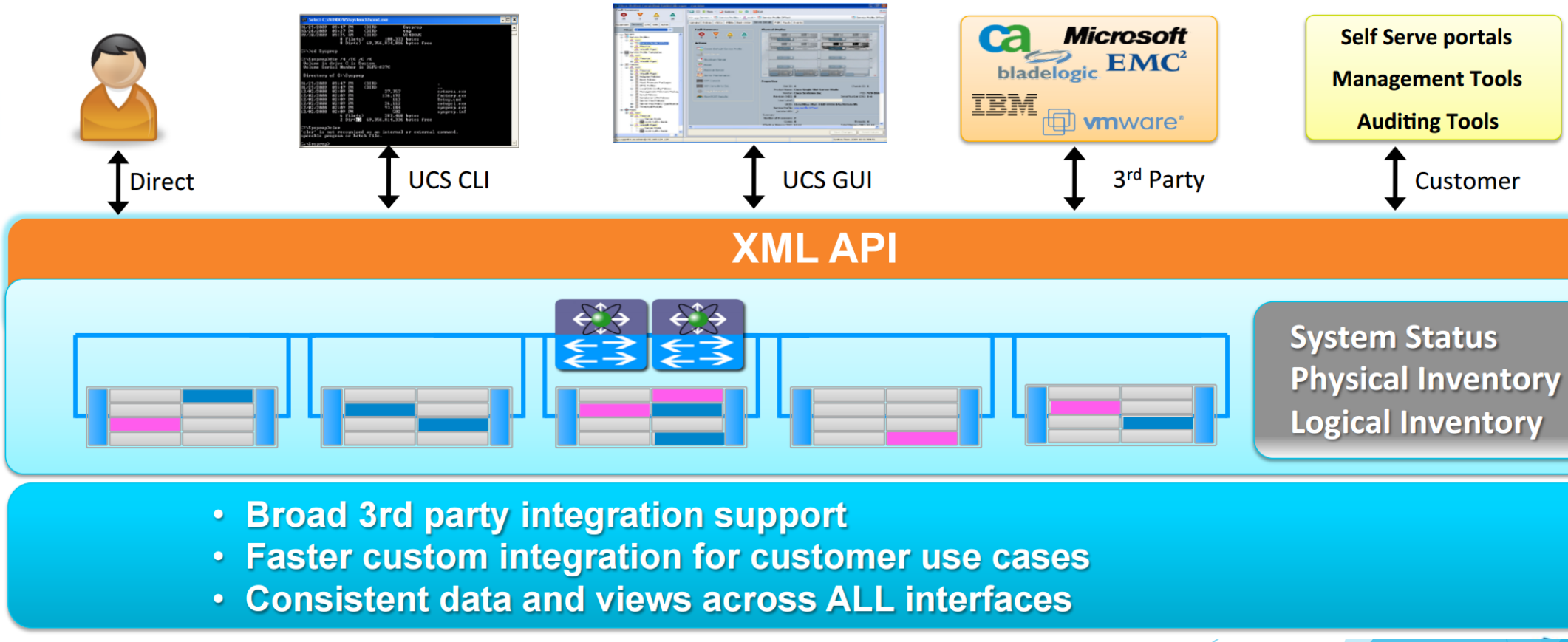


Attractive features of UCSM:

- 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



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?

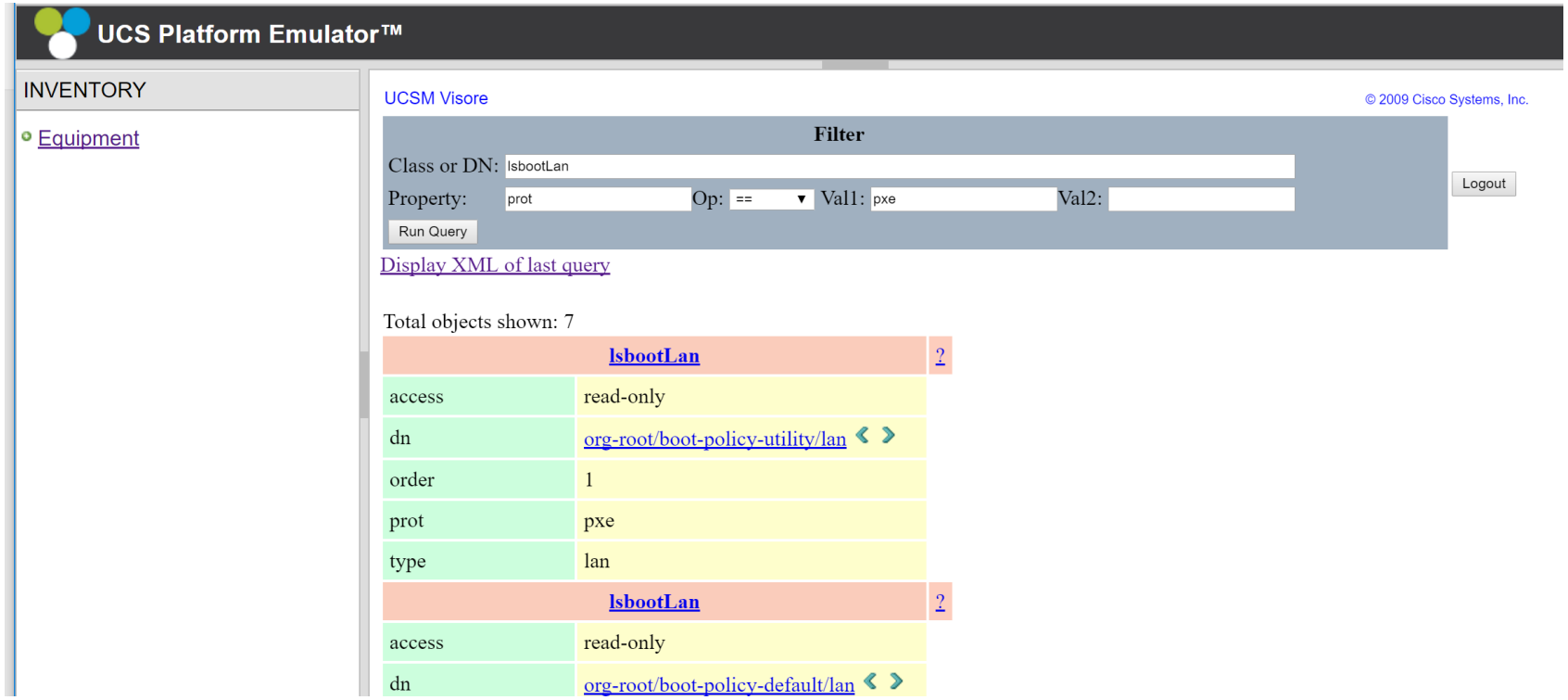
Every Managed Object is uniquely identified in the tree with its Distinguished Name (Dn) and can be uniquely identified within the context of its parent with its Relative Name (Rn).

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



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()

```
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

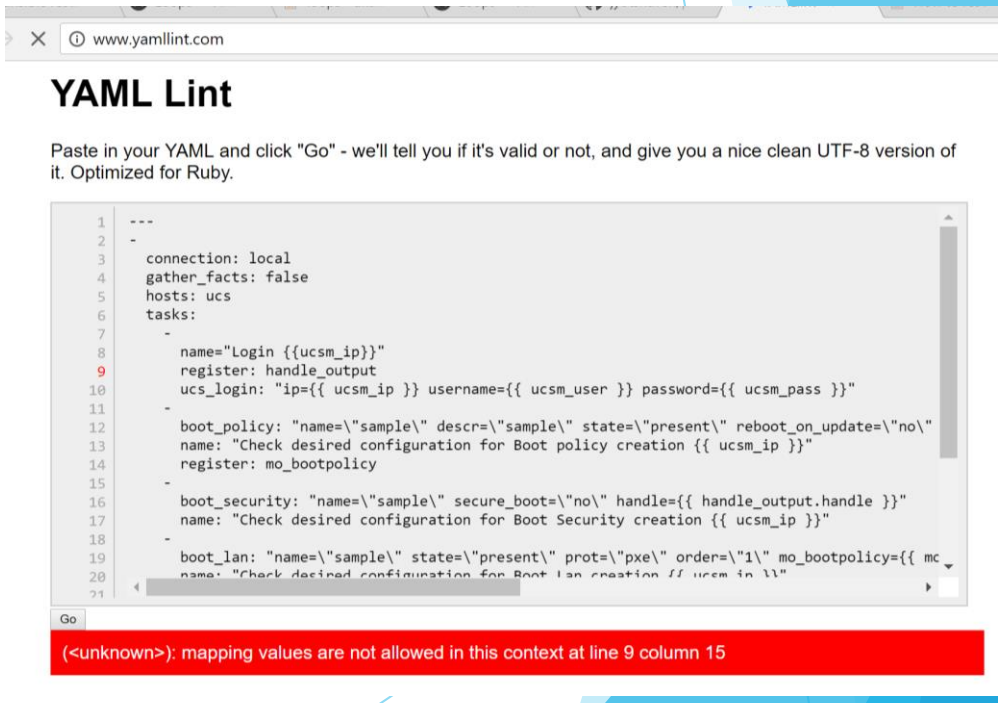
- Configuration management and orchestration
- What is a playbook?
- Each **playbook** contains one or more plays, which map hosts to a certain function.
- **Ansible** does this through something called tasks, which are basically module calls.

```
---
- 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



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

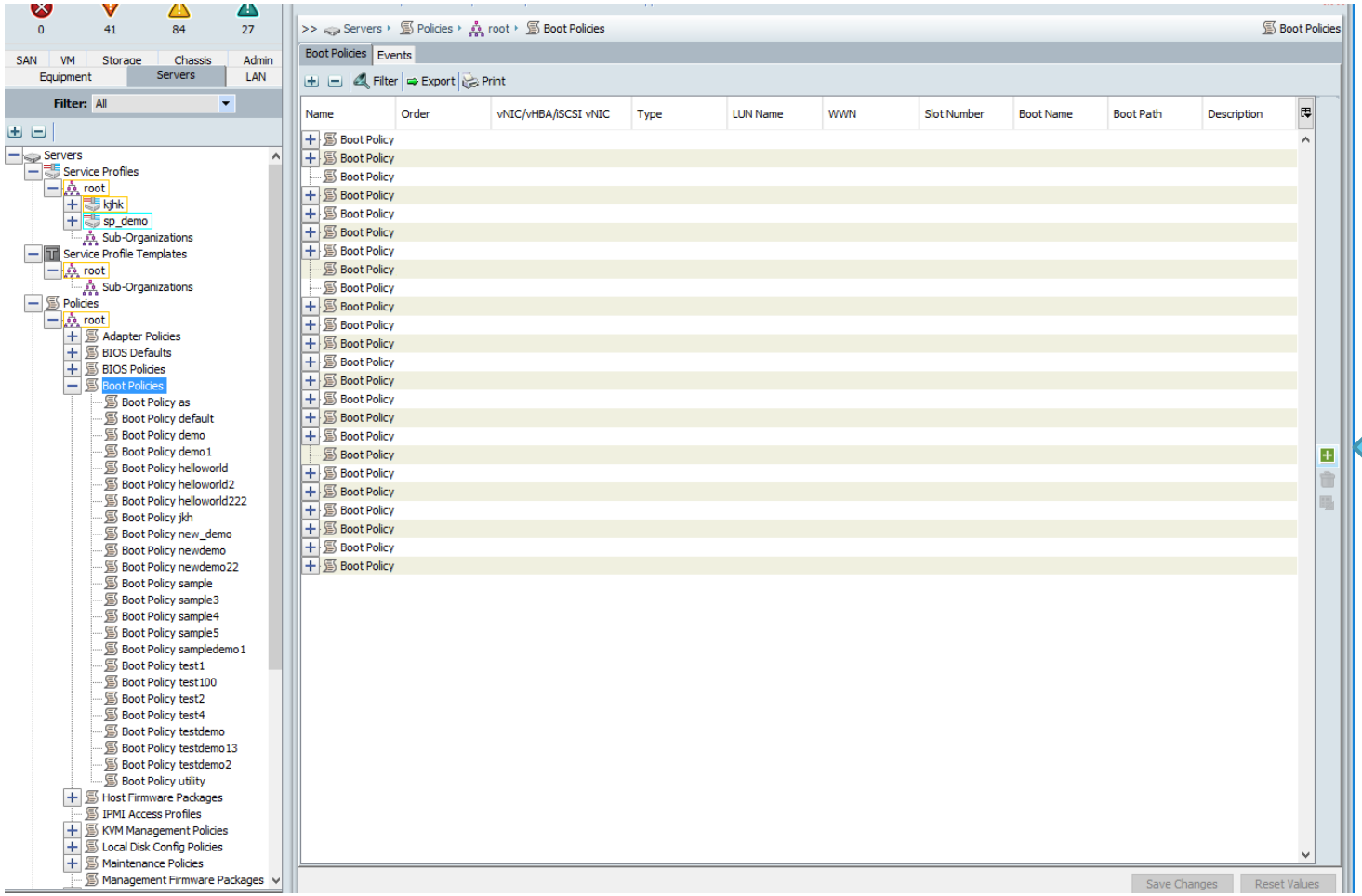
- module: boot_lan_vnic
- Short description: Create, modify or remove boot lan vnic primary and secondary
- description:
- Allows to check if boot lan vnic mo exists. If present, add a secondary boot lan vnic mo. If boot lan vnic mo is not present, create a primary boot lan vnic mo. If the desired state 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

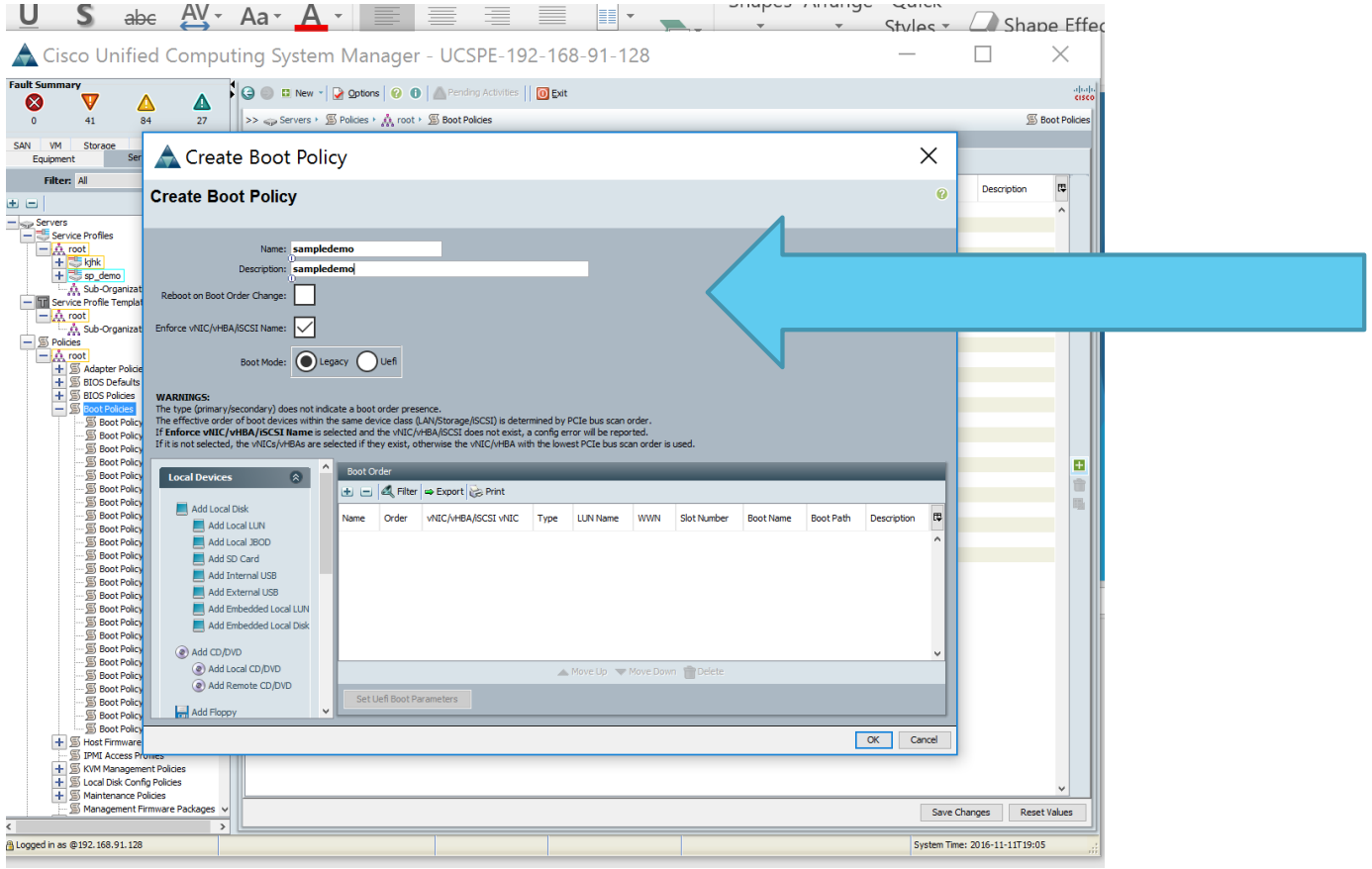
- Step 1:



Navigate to Servers tab. Click on this icon

Boot policy creation UI

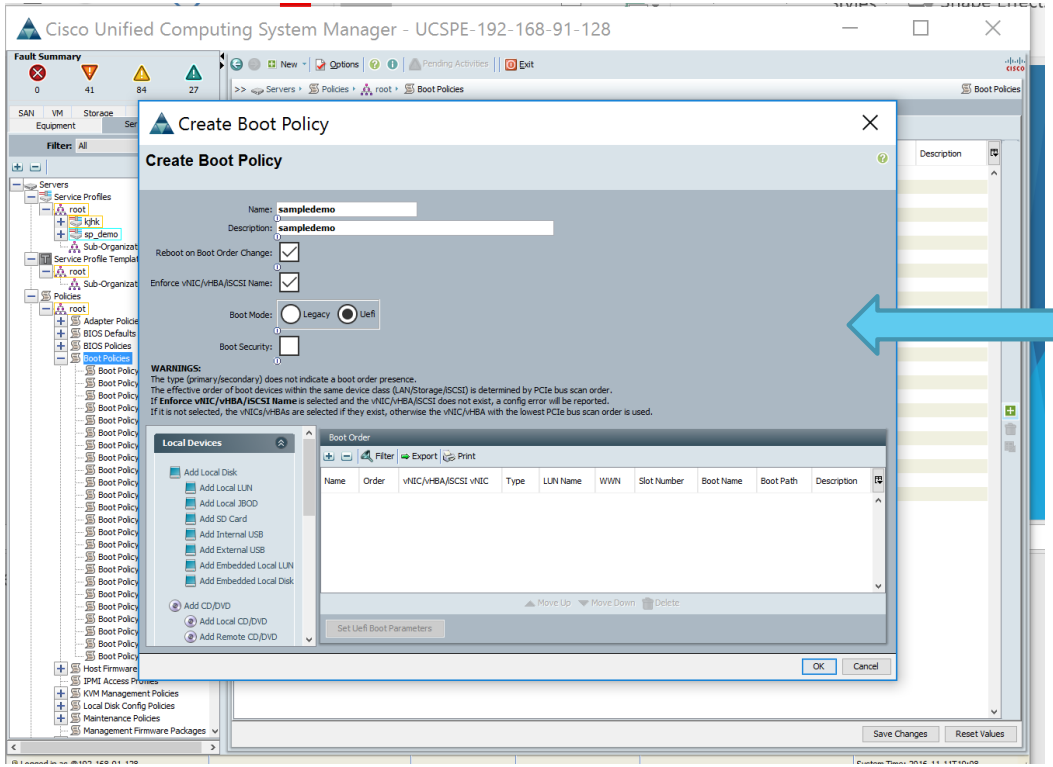
- Step 2:



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

Boot policy creation UI

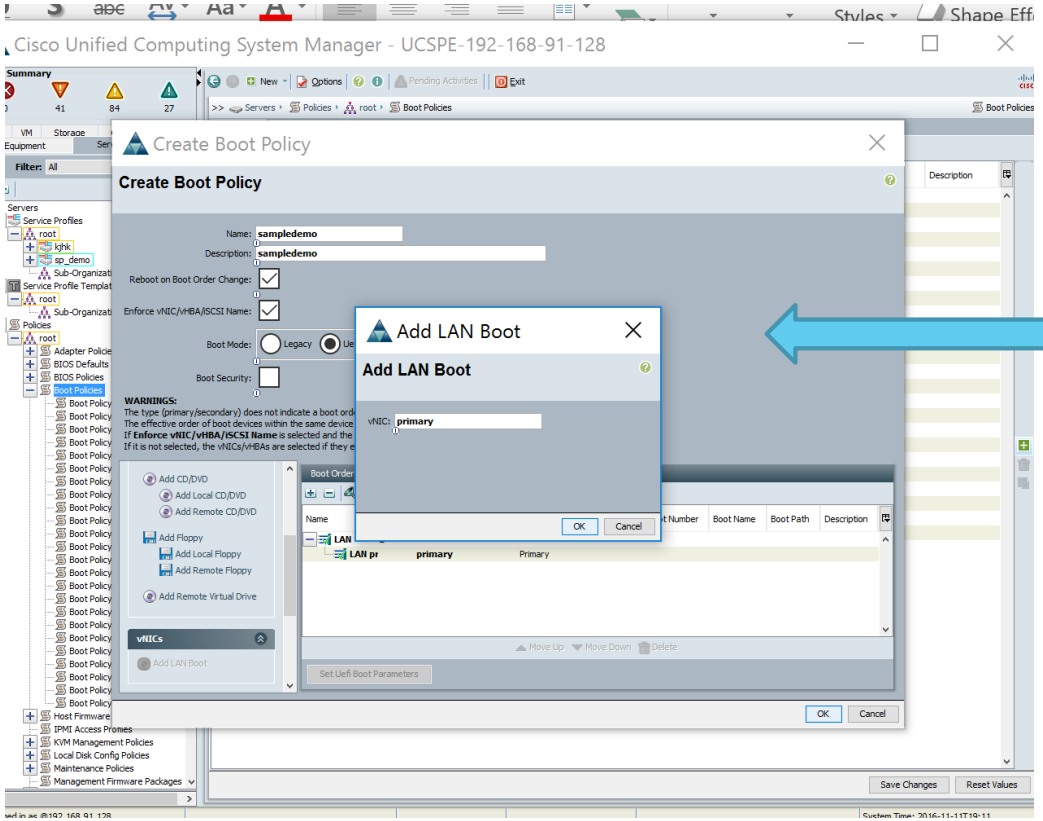
- Step 3:



Boot security becomes visible only for boot_mode option “uefi”

Boot policy creation UI

- Step 4:



Add boot_lan_vnic to boot order.

- Step 5: Save Changes and click OK!

Credits

This project is a brainchild of:

- David Soper (dsoper@cisco.com)
- Vikrant Balyan (vvb@cisco.com)

Please refer to a proof of concept by Vikrant at https://github.com/vvb/ucsm-ansible

I would like to thank David, Vikrant and Jose Delgado (jose@kloudxp.com) for their support and mentorship during this project!