Cisco Virtual Topology System (VTS) 2.6 Developer Guide

Updated: November 28, 2017

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Abstract

The Cisco Virtual Topology System (VTS) 2.6 Developer Guide gives information on VTS APIs and development features.

The Virtual Topology System (VTS) provides L2 and L3 connectivity to tenant, router, and service VMs. The main components of the VTS are the Virtual Topology Controller (VTC) and the XRVR.

Cisco Virtual Topology System 2.6 Developer Guide

© 1999–2017 Cisco Systems, Inc. All rights reserved.

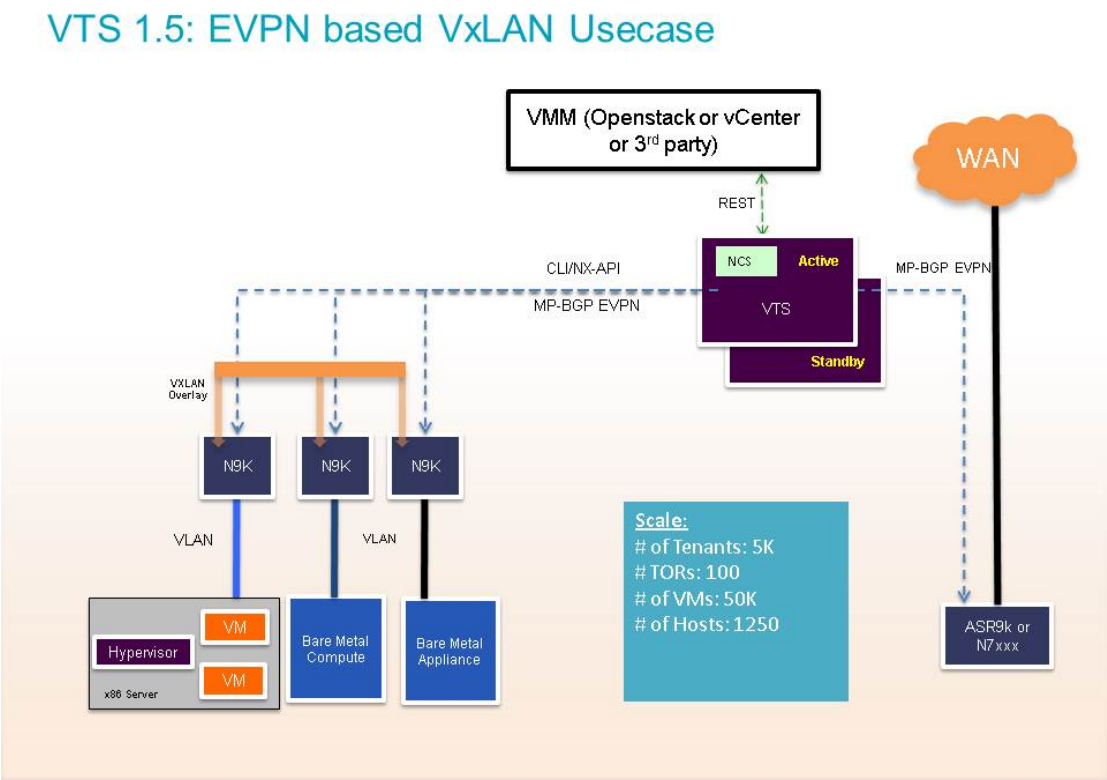
1 Contents

2 Overview of VTS 2.6 API

The VTS 2.1 API is exposed north-bound by the VTC component. The API is used to:

- Provision VTC with the Inventory of the DC topology
- Provision VTC with the Hardware VTEP Info
- Provision VTC with the information about the networks that will be serviced by the associated tenants
- Provision VTC with the information about sub-networks
- Provision VTC with the information about VMs attached to the networks
- Provision VTC with the information of L3 router and its interfaces that will be used for L3 connectivity between the tenant VMs

2.1 Architecture



2.1.1 Virtual Topology Controller (VTC)

VTC is the topology controller for the VTS; VTC controls all the hardware VTEPs to provide intra DC connectivity between VMs belonging to the same tenant zone in the DC and inter DC/WAN connectivity to remote networks belonging to the same tenant. VTC learns about remote networks by peering using BGP and other routing protocols to remote peers and route reflectors.

The VTC VM will have one VNIC interface which has VIP address:

- One for control, used in the NB API to program VTC.

2.2 Key Concepts

The VTC northbound API is a REST API over HTTPS. A VM managing entity is assumed to create VMs and call the VTC APIs to provide the required information for the creation of the virtual networks.
For security reasons the API implements certificate-based validation both on client side (the Managing Entity) and server side (VTC). The user is responsible for creating and uploading the server-side certificate in VTC, as well as creating and using the certificate on the client side. Instructions for the upload of certificate are specified in section 4.4.

3 Preparing to Use the VTS 2.6 API

The VTC NB API is used to provision the VTC with all hardware VTEPS that it controls, as well the networks, subnets and VMs that will be deployed on the compute servers.

3.1 Requirements

In order to exercise the API you should be able to issue https requests to VTC. This can be done by a binary or script. This document provides client-side examples using the curl command.

3.2 Installing the VTS 2.6 API

The VTS 2.6 API uses server-side certificate for security. The VTS 2.6 comes with default server-side certificate. The user is responsible for creating and replacing such certificates if needed.

3.2.1 Installing Certificates

- Key-file: Specifies which file that contains the private key for the certificate
  Copy the key file to /etc/ncs/ssl/cert/host.key

- Cert-file: Specifies which file that contains the server certificate.
  Copy the cert file to /etc/ncs/ssl/cert/host.cert

- restart the VTS server process:
  `sudo service ncs restart`

3.2.2 REST Semantics

- ***Retrieving a Resource***
  GET is used to retrieve a representation of a known resource.
  The XML representation has the list of container name as the XML document root.
  Request query parameters
  "deep" : retrieve a resource with all subresources incline
  "shallow" : retrieve a resource with no subresources incline
  "offset"/"limit" : used to specify a limited set of list entries to retrieve
  "select" : used to select which nodes and subresources in a resource to retrieve
- ***Replace a resource***
  PUT is used to completely replace a known resource.
- ***Update some properties of a resource***
  PATCH RFC 5789 is used to edit a known resource. PATCH cannot be used to change keys of a resource.
- ***Create a Resource***
  POST to the parent resource to create a new resource.
- ***DELETE a Resource***
  DELETE is used to remove a known resource. DELETE can be used to remove the entire resource or part of it.

4 API Reference

Base URI

https://<vtc-ip>:8888/api/running/cisco-vts/

Headers: All headers Content-Type, Accept and Authorization should be provided

Content-Type: application/vnd.yang.data+json

OR

Content-Type: application/vnd.yang.data+xml

Accept: application/vnd.yang.data+json

OR

Accept: application/vnd.yang.data+xml

Authorization: Basic: (username/password)

4.1 Authgroup

The following APIs are used to create or retrieve the authentication groups for

which the username and password that will be used to access the devices are stored.

GET
https://host:8888/api/running/devices/authgroups - to get list of authgroups.

PATCH
https://host:8888/api/running/devices/authgroups/ - to incrementally add the authgroup.

4.1.1 Payload

```
<authgroups>
<group>
<name>default_device</name>
<umap>
<local-user>admin</local-user><!-- -- Username used for NCS API authentication-- -- >
<remote-name>admin</remote-name><!-- -- Username for Device authentication-- -- >
<remote-password>admin</remote-password><!-- -- Password on Device for user account supplied-- -- >
<remote-secondary-password>admin</remote-secondary-password><!-- -- used for enable password-- -- >
</umap>
</group>
</authgroups>
```

4.2 Discovery

The following APIs are used to do discovery operation for the devices, fabric connections and hosts.

4.2.1 Initiate a discovery process

Initiates the backend discovery process
https://localhost:8443/VTS/rs/vtsService/discovery
Http method: POST

4.2.1.1 Payload

```
{
  "discovery": {
    "input": {
      "seedDeviceIp": "10.104.244.89",
      "seedDeviceUserName": "admin",
      "seedDevicePassword": "Cisco123!"
    }
  }
}
```

4.2.1.2 Response


```
{
 "discovery": {
  "status": "IN_PROGRESS",
  "status_msg": "",
  "errors": [

  ],
  "discovery_time": "2 seconds",
  "time": "2017-07-17 06:13:16.436103"
 }
```


4.2.2 Query discovery status

To check the status of discovery process
https://localhost:8443/VTS/rs/vtsService/discovery
Http method: GET

4.2.2.1 Response


```
{
 "discovery": {
  "status": "END",
  "status_msg": "Success",
  "errors": [

  ],
  "discovery_time": "2 seconds",
  "time": "2017-07-17 06:13:19.316523"
 }
}
```


4.2.3 Get discovered data

This API will execute and return the final JSON data based on one parameter i.e. **reconcile**.

4.2.3.1 Reconcile as true

If the value of **reconcile** is true it will return reconciled data i.e. it will compare the existing inventory and new inventory and give result in new , mismatch , existing and missing bucket .

https://localhost:8443/VTS/rs/vtsService/discovery/details/reconcile=true
Http method: GET

4.2.3.1.1 Response


```
{
  "reconciled_inventory_details": {
    "devices": {
      "newItems": [
        {
          "name": "tb55-spine",
          "deviceIp": "10.104.244.87",
          "authGroup": "TB55",
```

```
            "devicePlatform": "N9k",
            "deviceRole": "spine-rr",
            "groupTag": "",
            "bgpasn": null,
            "loopbackInterfaceNumber": null,
            "workloadDetails": null
         },
         {
            "name": "tb55-tor1",
            "deviceIp": "10.104.244.85",
            "authGroup": "TB55",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "bgpasn": null,
            "loopbackInterfaceNumber": null,
            "workloadDetails": null
         },
         {
            "name": "tb55-tor3",
            "deviceIp": "10.104.244.88",
            "authGroup": "TB55",
            "devicePlatform": "N7k",
            "deviceRole": "leaf",
            "groupTag": "",
            "bgpasn": null,
            "loopbackInterfaceNumber": null,
            "workloadDetails": null
         },
         {
            "name": "tb55-tor2",
            "deviceIp": "10.104.244.86",
            "authGroup": "TB55",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "bgpasn": null,
            "loopbackInterfaceNumber": null,
            "workloadDetails": null
         }
      ],
      "mismatched": [],
      "existing": [],
      "missing": []
   },
   "fabricInterfaces": {
      "newItems": [
         {
            "sourceDeviceName": "tb55-tor3",
            "sourceDeviceInterface": "Ethernet1/50",
            "destinationDeviceName": "tb55-spine",
            "destinationInterfaceName": "Ethernet1/55",
            "destinationDeviceIP": null,
            "deviceType": "baremetal"
         },
         {
            "sourceDeviceName": "tb55-spine",
            "sourceDeviceInterface": "Ethernet1/50",
            "destinationDeviceName": "tb55-tor2",
            "destinationInterfaceName": "Ethernet1/50",
            "destinationDeviceIP": null,
            "deviceType": "baremetal"
         },
         {
            "sourceDeviceName": "tb55-tor1",
            "sourceDeviceInterface": "Ethernet1/49",
            "destinationDeviceName": "tb55-spine",
```

```
            "destinationInterfaceName": "Ethernet1/49",
            "destinationDeviceIP": null,
            "deviceType": "baremetal"
         },
         {
            "sourceDeviceName": "tb55-spine",
            "sourceDeviceInterface": "Ethernet1/49",
            "destinationDeviceName": "tb55-tor1",
            "destinationInterfaceName": "Ethernet1/49",
            "destinationDeviceIP": null,
            "deviceType": "baremetal"
         },
         {
            "sourceDeviceName": "tb55-tor2",
            "sourceDeviceInterface": "Ethernet1/50",
            "destinationDeviceName": "tb55-spine",
            "destinationInterfaceName": "Ethernet1/50",
            "destinationDeviceIP": null,
            "deviceType": "baremetal"
         }
      ],
      "mismatched": [],
      "existing": [],
      "missing": []
   },
   "hosts": {
      "newItems": [
         {
            "hostName": "tb55-compute-84",
            "hostIP": "10.104.244.48",
            "vmmID": null,
            "virtualSwitch": null,
            "workloadDetails": null,
            "hostType": "virtual-server",
            "hostInterfaces": {
               "newItems": [
                  {
                     "hostName": "tb55-compute-84",
                     "hostInterface": "eth3",
                     "sriovEnabled": true,
                     "physNet": "",
                     "attachedToDevice": "tb55-tor2",
                     "deviceInterface": "Ethernet1/4"
                  }
               ],
               "mismatched": [],
               "existing": [],
               "missing": []
            }
         },
         {
            "hostName": "tb55-controller-81",
            "hostIP": "10.104.244.81",
            "vmmID": null,
            "virtualSwitch": null,
            "workloadDetails": null,
            "hostType": "virtual-server",
            "hostInterfaces": {
               "newItems": [
                  {
                     "hostName": "tb55-controller-81",
                     "hostInterface": "eth3",
                     "sriovEnabled": false,
                     "physNet": "",
                     "attachedToDevice": "tb55-tor1",
                     "deviceInterface": "Ethernet1/1"
                  }
```

```
                ],
                "mismatched": [],
                "existing": [],
                "missing": []
            }
        },
        {
            "hostName": "tb55-compute-83",
            "hostIP": "10.104.244.83",
            "vmmID": null,
            "virtualSwitch": null,
            "workloadDetails": null,
            "hostType": "virtual-server",
            "hostInterfaces": {
                "newItems": [
                    {
                        "hostName": "tb55-compute-83",
                        "hostInterface": "eth3",
                        "sriovEnabled": false,
                        "physNet": "",
                        "attachedToDevice": "tb55-tor2",
                        "deviceInterface": "Ethernet1/3"
                    }
                ],
                "mismatched": [],
                "existing": [],
                "missing": []
            }
        },
        {
            "hostName": "tb55-compute-82",
            "hostIP": "10.104.244.82",
            "vmmID": null,
            "virtualSwitch": null,
            "workloadDetails": null,
            "hostType": "virtual-server",
            "hostInterfaces": {
                "newItems": [
                    {
                        "hostName": "tb55-compute-82",
                        "hostInterface": "eth3",
                        "sriovEnabled": false,
                        "physNet": "",
                        "attachedToDevice": "tb55-tor1",
                        "deviceInterface": "Ethernet1/2"
                    }
                ],
                "mismatched": [],
                "existing": [],
                "missing": []
            }
        }
    ],
    "mismatched": [],
    "existing": [],
    "missing": []
  }
 }
}
```


4.2.3.2 Reconcile as false

If the value of **reconcile** is false it will return discovered data.

https://localhost:8443/VTS/rs/vtsService/discovery/details/reconcile=false

Http method: GET

4.2.3.2.1 Response


```
{
   "resource": {
      "deviceDetails": [
         {
            "deviceName": "tb55-tor1",
            "deviceIp": "10.104.244.85",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "portName": "Ethernet1/1",
            "connectionType": "server",
            "serverId": "tb55-controller-81",
            "serverType": "virtual-server",
            "interfaceName": "eth3",
            "serverIp": "10.104.244.81",
            "authGroup": "TB55",
            "sriovEnabled": "FALSE",
            "physnetName": ""
         },
         {
            "deviceName": "tb55-tor1",
            "deviceIp": "10.104.244.85",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "portName": "Ethernet1/2",
            "connectionType": "server",
            "serverId": "tb55-compute-82",
            "serverType": "virtual-server",
            "interfaceName": "eth3",
            "serverIp": "10.104.244.82",
            "authGroup": "TB55",
            "sriovEnabled": "FALSE",
            "physnetName": ""
         },
         {
            "deviceName": "tb55-tor1",
            "deviceIp": "10.104.244.85",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "portName": "Ethernet1/49",
            "connectionType": "fabric",
            "serverId": "tb55-spine",
            "serverType": "baremetal",
            "interfaceName": "Ethernet1/49",
            "serverIp": "10.104.244.87",
            "authGroup": "TB55",
            "sriovEnabled": "FALSE",
            "physnetName": ""
         },
         {
            "deviceName": "tb55-tor2",
            "deviceIp": "10.104.244.86",
            "devicePlatform": "N9k",
            "deviceRole": "leaf",
            "groupTag": "",
            "portName": "Ethernet1/3",
            "connectionType": "server",
            "serverId": "tb55-compute-83",
            "serverType": "virtual-server",
            "interfaceName": "eth3",
            "serverIp": "10.104.244.83",
```

```
    "authGroup": "TB55",
    "sriovEnabled": "FALSE",
    "physnetName": ""
  },
  {
    "deviceName": "tb55-tor2",
    "deviceIp": "10.104.244.86",
    "devicePlatform": "N9k",
    "deviceRole": "leaf",
    "groupTag": "",
    "portName": "Ethernet1/4",
    "connectionType": "server",
    "serverId": "tb55-compute-84",
    "serverType": "virtual-server",
    "interfaceName": "eth3",
    "serverIp": "10.104.244.48",
    "authGroup": "TB55",
    "sriovEnabled": "TRUE",
    "physnetName": ""
  },
  {
    "deviceName": "tb55-tor2",
    "deviceIp": "10.104.244.86",
    "devicePlatform": "N9k",
    "deviceRole": "leaf",
    "groupTag": "",
    "portName": "Ethernet1/50",
    "connectionType": "fabric",
    "serverId": "tb55-spine",
    "serverType": "baremetal",
    "interfaceName": "Ethernet1/50",
    "serverIp": "10.104.244.87",
    "authGroup": "TB55",
    "sriovEnabled": "FALSE",
    "physnetName": ""
  },
  {
    "deviceName": "tb55-spine",
    "deviceIp": "10.104.244.87",
    "devicePlatform": "N9k",
    "deviceRole": "spine-rr",
    "groupTag": "",
    "portName": "Ethernet1/49",
    "connectionType": "fabric",
    "serverId": "tb55-tor1",
    "serverType": "baremetal",
    "interfaceName": "Ethernet1/49",
    "serverIp": "10.104.244.85",
    "authGroup": "TB55",
    "sriovEnabled": "FALSE",
    "physnetName": ""
  },
  {
    "deviceName": "tb55-spine",
    "deviceIp": "10.104.244.87",
    "devicePlatform": "N9k",
    "deviceRole": "spine-rr",
    "groupTag": "",
    "portName": "Ethernet1/50",
    "connectionType": "fabric",
    "serverId": "tb55-tor2",
    "serverType": "baremetal",
    "interfaceName": "Ethernet1/50",
    "serverIp": "10.104.244.86",
    "authGroup": "TB55",
    "sriovEnabled": "FALSE",
    "physnetName": ""
```

```
        },
        {
          "deviceName": "tb55-tor3",
          "deviceIp": "10.104.244.88",
          "devicePlatform": "N7k",
          "deviceRole": "leaf",
          "groupTag": "",
          "portName": "Ethernet1/50",
          "connectionType": "fabric",
          "serverId": "tb55-spine",
          "serverType": "baremetal",
          "interfaceName": "Ethernet1/55",
          "serverIp": "10.104.244.87",
          "authGroup": "TB55",
          "sriovEnabled": "FALSE",
          "physnetName": ""
        }
      ]
    }
}
```


4.3 Device

The following APIs are used to do CRUD operation of the device.
GET
https://host:8888/api/running/devices/device - to get list of devices

PUT
https://host:8888/api/running/devices/device/{device-name} - to add device to VTS or replace a device

PATCH
https://host:8888/api/running/devices/device/ - to incrementally add the device (Existing device will not be modified)

4.3.1 Payload to add N9K device

```
<device>
  <name>n9k1</name>
  <address>1.1.1.1</address>
  <authgroup>N9k-authgroup1</authgroup>
  <device-type>
   <cli>
    <ned-id>cisco-nx</ned-id>
    <protocol>ssh</protocol>
   </cli>
  </device-type>
  <ned-settings>
<cisco-nx-connection>
        <method>nxapi</method>
</cisco-nx-connection>
  </ned-settings>
```

```
        <state>
          <admin-state>unlocked</admin-state>
          <admin-state-description>changed to xyz</admin-state-description>
        </state>
         <device-info>
              <device-use>LEAF</device-use>
              <bgp-peering>
                   <bgp-asn>100</bgp-asn>
                   <loopback-if-num>1</loopback-if-num>
                   <loopback-if-ip>10.10.10.0/24</loopback-if-ip>
              </bgp-peering>
         </device-info>
       </device>
```

4.4 Inventory

The following APIs are used to upload / retrieve the inventory of the topology.

GET

https://host:8888/api/running/cisco-vts/devices to get list of device and the connected servers information on each device

PUT

https://host:8888/api/running/cisco-vts/devices/device/{device-name} - to add deviceport-server info to NCS or replace a deviceport-server info for this device

PATCH

https://host:8888/api/running/cisco-vts/devices - to incrementally add the deviceport-server info (Existing device will not be modified)

4.4.1 Payload Example

```
<devices
   xmlns="http://cisco.com/ns/yang/vts"
   xmlns:y="http://tail-f.com/ns/rest"
   xmlns:vts="http://cisco.com/ns/yang/vts">
   <device>
      <name>tb3-tor1</name>
      <ports>
         <port>
            <name>Ethernet1/1</name>
            <connection-type
               xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">cisco-vts-identities:server
            </connection-type>
            <servers>
               <server>
                  <name>controller</name>
                  <type
                     xmlns:cisco-vts-identities="http://cisco.com/ns/yang/vts/identities">
```

```
                    cisco-vts-identities:virtual-server
                  </type>
                  <interface-name>eth3</interface-name>
                  <ip>172.20.100.30</ip>
                  <connid>e60792bc-35a3-4785-89ac-55865a0fd237</connid>
                </server>
              </servers>
            </port>
            <port>
              <name>Ethernet1/2</name>
              <connection-type
                xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">cisco-vts-identities:server
              </connection-type>
              <servers>
                <server>
                  <name>compute1</name>
                  <type
                    xmlns:cisco-vts-identities="http://cisco.com/ns/yang/vts/identities">
                    cisco-vts-identities:virtual-server
                  </type>
                  <interface-name>eth3</interface-name>
                  <ip>172.20.100.31</ip>
                  <connid>9d45feea-770e-4d16-80f7-174e41947636</connid>
                </server>
              </servers>
            </port>
          </ports>
    </device>
</devices>
```

4.4.2 Payload Example with FEX

```
<devices
    xmlns="http://cisco.com/ns/yang/vts"
    xmlns:y="http://tail-f.com/ns/rest"
    xmlns:vts="http://cisco.com/ns/yang/vts">
    <device>
       <name>tb3-tor1</name>
 <ports>
           <port>
              <name>port-channel25</name>
              <connection-type
                xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">cisco-vts-identities:fabric
              </connection-type>
              <fexs>
                <fex>
                  <fex-id>120</fex-id>
                  <ports>
                    <port>
```

```
                        <name>Ethernet120/1/1</name>
                        <connection-type
                           xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">
                           cisco-vts-identities:server
                        </connection-type>
                        <connid>0b4451cf-152e-490c-82f9-e39ffadf29c4</connid>
                        <servers>
                           <server>
                              <name>1.0.1.3</name>
                              <type
                                 xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">
                              cisco-vts-identities:virtual-server
                              </type>
                              <interface-name>Ethernet1/7</interface-name>
                              <ip>1.0.0.10</ip>
                              <connid>77e76cbe-13e0-4a30-b921-
7acfb6d67ac5</connid>
                           </server>
                     </servers>
                      </port>
                    </ports>
                  </fex>
               </fexs>
            </port>
         </ports>
      </device>
</devices>
```

For VTS2.5.1, two fields have been added additionally to the hosts.

1.) virtual-switch
2.) vtf-link

Parameter	Value	VMM-ID/Desciption
virtual-switch	not-defined-st ovs-st dvs-st vtf-vtep-st vtf-l2-st	Openstack/vCenter Openstack vCenter Openstack/vCenter Openstack
vtf-link	vtf-tor-link vtf-vtsr-link	Uplink tor connection Vtsr connection

4.5 Example payload for virtual switch type and vtf-link.

```
<devices
   xmlns="http://cisco.com/ns/yang/vts"
   xmlns:y="http://tail-f.com/ns/rest"
   xmlns:vts="http://cisco.com/ns/yang/vts">
   <device>
      <name>tb3-tor1</name>
      <ports>
         <port>
            <name>Ethernet1/1</name>
            <connection-type
               xmlns:cisco-vts-
identities="http://cisco.com/ns/yang/vts/identities">cisco-vts-identities:server
            </connection-type>
            <servers>
               <server>
                  <name>controller</name>
                  <type
                     xmlns:cisco-vts-identities="http://cisco.com/ns/yang/vts/identities">
                     cisco-vts-identities:virtual-server
                  </type>
                  <interface-name>eth3</interface-name>
                  <ip>172.20.100.30</ip>
                  <virtual-switch> ovs-st </virtual-switch>
                  <vtf-link> vtf-tor-link </vtf-link>
                  <connid>e60792bc-35a3-4785-89ac-55865a0fd237</connid>
               </server>
            </servers>
         </port>
      </ports>
   </device>
</devices>
```

Configuring Tenant Information

https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<name>

Tenants	Container for set of VTS Tenants
Tenant	List of VTS Tenants
name	Tenant Name

Configuring Topology Information

https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<id>

Topologies	Container for set of Topologies belonging to a tenant
Topology	List of Topologies
id	Topology Name

4.6 Device-Groups

A device-group can be used to group a set of L2/L3 VTEP devices such that these share a common vlan pool. The VTEPs can be physical or virtual end points. Once added to the device-group, the device specific vlan pools are no longer used.

Operations: CREATE/UPDATE/GET/DELETE

Create (PUT)

Create a device-group in VTS
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

Parameter	Value	Description
Name	String	Name of the device group
Type	String	Type of device group. Default “device-group-type”
vmmid	String	VMM ID
devices	-	Container for list of devices
device/name	String	Name of device in VTS inventory
device/type	String	Type of device – physical or virtual – p-vtep/v-vtep

Sample Payload

https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/tb12-tor-group

```
"device-group": [
      {
        "name": "tb12-tor-group"
        "type": "device-group-type",
        "vmmid": "6BA0C760-E840-40A4-A949-B06DD18E0998",
        "devices": {
          "device": [
            {
              "name": "tb12-leaf1",
               "type": "p-vtep"
            },
            {
              "name": "tb12-leaf2",
               "type": "p-vtep"
            }
          ]
        }
      }
  ]
```

Update (PATCH)
Create a device-group in VTS
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group

```
"device-group": [
      {
        "name": "tb12-tor-group"
        "type": "device-group-type",
        "vmmid": "6BA0C760-E840-40A4-A949-B06DD18E0998",
        "devices": {
          "device": [
            {
              "name": "tb12-leaf1",
          "type": "p-vtep"
            },
            {
              "name": "tb12-leaf2",
              "type": "p-vtep"
            },
                {
                  “name”:”tb12-virtual-leaf1”,
              “type”:”v-vtep”
            }
          ]
        }
      }
  ]
```

GET
Show device-group details
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

LIST (GET)
Show list of all device-groups in the system
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/

DELETE
Delete a device group.
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

4.7 Device-Interface-Groups

A device-interface-group can be used to group a set of Physical Interface of Nexus 7000 devices so that user can use common dot1Q pool for Overlay Network provisioning. VTS automatically creates a system defined group per interface during inventory upload and users can use this API to control the group members

Operations: CREATE/UPDATE/GET/DELETE

Create (PUT)

Create a device-interface-group in VTS
https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/<device-interface-group-name>

Parameter	Value	Description
Name	String	Name of the device Interface group
Type	String	Type of device Interface group. Default "device-interface-group"
Is User Defined Group	Boolean	Flag to determine if the group is user defined or system defined
device-interfaces	-	Container for list of device interfaces
device-interfaces/device-interface	String	device interfaces from devices
device-interface/name	String	Name of the device interface
device-interface/device-name	String	Name of the device the interface belongs to
Device-interface/pool-name	String	Device interface pool this group uses

Sample Payload

https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/tb12-interface-group


```
{
  "device-interface-group":{
    "name":"tb12-interface-group",
    "type":"cisco-vts-identities:device-interface-group",
    "is-user-defined-group":true,
    "device-interfaces":{
      "device-interface":[
        {
          "name":"Ethernet1\/1",
          "device-name":"n7k-1-JenShue-VDC1",
          "pool-name":" tb12-interface-pool "
        },
        {
          "name":"Ethernet1\/2",
          "device-name":"n7k-1-JenShue-VDC1",
          "pool-name":" tb12-interface-pool "
        },
        {
          "name":"Ethernet1\/3",
          "device-name":"n7k-2-JenShue-VDC1",
```

```
            "pool-name":"tb12-interface-pool"
          }
        ]
      }
    }
}
```


Update (PATCH)

Update a device-interface-group in VTS

https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group

https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/tb12-interface-group


```
"device-interface-group": [
  {
    "name":"tb12-interface-group",
    "type":"cisco-vts-identities:device-interface-group",
    "is-user-defined-group":true,
    "device-interfaces":{
      "device-interface":[
        {
          "name":"Ethernet1\/1",
          "device-name":"n7k-1-JenShue-VDC1",
          "pool-name":" tb12-interface-pool "
        },
        {
          "name":"Ethernet1\/2",
          "device-name":"n7k-1-JenShue-VDC1",
          "pool-name":" tb12-interface-pool "
        },
        {
          "name":"Ethernet1\/3",
          "device-name":"n7k-2-JenShue-VDC1",
          "pool-name":"tb12-interface-pool"
        }
      ]
    }
  }
]
```


GET

Show device-interface-group details

https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/<device-interface-group-name>

LIST (GET)

Show list of all device-interface-groups in the system

https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/

DELETE
Delete a device interface group.
https://172.20.100.126:8888/api/running/cisco-vts/device-interface-groups/device-interface-group/<device-interface-group-name>

4.8 Network Extensions

Allows to manipulate network extension for a tenant. Settings changed here would affect all the networks that are inheriting the extension, under the tenant. extend the layer 2 network to core.

Create (PUT)
Creates Network Extensions
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/network-extensions

Payload

Parameter	Value	Description
extend-networks-to-core	no/yes	Yes means extend all inheriting networks to core.

Sample Payload

```
{
  "network-extensions": {
    "extend-networks-to-core": "yes"
  }
}
```

GET
Show network Network Extensions under a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/network-extensions

DELETE
Delete Network Extensions under a tenant <tenant-name> and associated resources.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/network-extensions

4.9 Network

Shows/Lists information for creates, updates and deletes of Network. VTS supports IPv4 and IPv6 overlay networks

Create (PUT)

Create a network in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

Payload

Parameter	Value	Description
admin-state-up	true/false	Admin state of the network up or down
id	Uuid	Network ID
name	String	Network name
shared	Boolean	Shared across all tenants
status	active/build/down	Network Status
provider-network-type	gre/trunk/vlan/vxlan	Network Type
provider-physical-network	String	Physical Network Name
router-external	Boolean	External access to Network
extend-to-core	no/yes/none	Extend the l2 network to core. None or empty means the values in inherited from Tenant. yes or no means the value is overridden at network level.
Provider-segmentation-id	String	Segmentation ID
arp-suppression	String	Enable or Disable ARP suppression for Network. By default ARP suppression is Disabled.

Sample Payload

```
{
  "network":{
    "id":"14528457-1761-4378-bdaf-1e4b2ff5e999",
    "admin-state-up":true,
    "name":"network-1",
    "provider-physical-network":"physnet1",
    "provider-segmentation-id":"1111",
    "provider-network-type":"cisco-vts-identities:vxlan",
    "router-external":false,
    "extend-to-core":"none",
    "shared":false,
    "status":"cisco-vts-identities:active"
    "arp-suppression":"nwk-arp-suppression-disabled"
  }
}
```

GET

Show network details associated to one VTS Network under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

LIST (GET)

Show list of all networks under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/

DELETE

Delete Network under topology <topology-name> of a tenant <tenant-name> and associated resources.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

4.10 Subnet

Shows/Lists information for creates, updates and deletes of Subnet. VTS allows only one IPv4 and one IPv6 subnet per network

Create (PUT)

Create a subnet in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

Parameter	Value	Description
network-id	Uuid	Network ID
id	Uuid	Subnet ID
name	String	Subnet name
shared	Boolean	Shared across all tenants
status	active/build/down	Network Status
enable-dhcp	Boolean	True if DHCP is enabled
cidr	String	The CIDR
gateway-ip	String	The gateway IP address
ip-version	String	IP version 4 or 6
ipv6-ra-mode	String	IPv6 ra configuration mode
ipv6-address-mode	String	IPv6 address configuration mode

Sample Payload:

IPv4 subnet:

```
{
 "subnet":
  {
   "id": "98944325-eb56-56f1-9468-095028d8d36c", --
   "network-id": "489ed271-9427-4fa0-85ca-1ca7187d29d1",--
   "name": "subnet1",--
```

```
    "enable-dhcp": true,
    "cidr": "2.2.2.0/24",
    "gateway-ip": "2.2.2.1",
    "ip-version": "4",
    "shared": false --
      }
}
```


IPv6 Subnet:


```
{
 "subnet":
  {
   "id": "98944325-eb56-56f1-9468-095028d8d36c",
   "network-id": "489ed271-9427-4fa0-85ca-1ca7187d29d1",
   "name": "subnet6",
   "enable-dhcp": true,
   "cidr": "1111::0/64",
   "gateway-ip": "1111::1",
   "ip-version": "6",
   "shared": false
     }
}
```


GET

Show subnet details associated to one VTS Subnet under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

LIST (GET)

Show list of all subnets under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/

DELETE

Delete subnet under topology <topology-name> of a tenant <tenant-name> and associated resources.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

4.11 VM Port

Shows/Lists information for creates, updates and deletes of port

Create (PUT)

Create a port in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

Parameter	Value	Description
network-id	uuid	Network ID
id	uuid	port ID
name	String	Port name
status	cisco-vts-identities :active/cisco-vts-identities:build/cisco-vts-identities:down	Port status
admin-state-up	boolean	Admin state of the network up or down
connid	uuid	Connection id of the device
tagging	optional/mandatory	Indicates whether a baremetal server tags the packets it sends
binding-host-id	String	Host identifier.
type	cisco-vts-identities:baremetal/ cisco-vts-identities:virtual-server	Type of server port.
mac-address	mac-address string or unknown-{32,36}	Port Mac Address

Sample Payload:


```
{
  "port":{
    "id":"bb42b3c9-1515-4c23-9af7-bfcf1d11750d",
    "status":"cisco-vts-identities:active",
    "tagging":"optional",
    "network-id":"0deeba97-e36b-4e99-b43f-6df7d278655e",
    "binding-host-id":"alibaba-compute5",
    "connid":[
      {
        "id":"2ff81bb9-3b75-4f15-9ef5-6ad03b72cf2e"
      }
    ],
    "admin-state-up":"true",
    "type":"cisco-vts-identities:baremetal",
    "mac-address":"unknown-bb42b3c9-1515-4c23-9af7-bfcf1d11750d"
  }
}
```


GET

Show port details associated to one VTS Port under topology <topology-name> of a tenant <tenant-name>, belonging to network with id <id>.

https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

LIST (GET)
Show list of all ports under topology <topology-name> of a tenant <tenant-name> and network with id <id>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/

DELETE
Delete port under topology <topology-name> of a tenant <tenant-name> and network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

4.12 Trunk

Shows/Lists information for creates, updates and deletes of trunks.

Create (PUT)
Create a trunk in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/trunks/trunk/<id>

Parameter	Value	Description
id	String	Trunk ID
name	String	Trunk name
tenant-name	String	Tenant name
topology-id	String	Topology ID
port-id	String	ID of parent port
sub-ports	-	Container for list of subports
sub-ports/port-id	String	Port ID of subport
sub-ports/ segmentation-id	String	Segmentation ID of subport
sub-ports/ segmentation-type	String	Segmentation type of subport
sub-ports/mac-address	String	MAC address of subport

Sample Payload:

```
{
  "trunk": {
   "port-id": "afc235a1-92a5-423f-8093-b4d25652ab69",
   "tenant-name": "admin",
   "topology-id": "admin",
   "sub-ports": [
     {
       "port-id": "5688b4de-e26a-4e66-9542-3c2b47809ea9",
       "segmentation-type": "cisco-vts-identities:vlan",
       "segmentation-id": 1001,
```

```
      "mac-address": "fa:16:3e:f6:8f:2e"
     },
     {
      "port-id": "d88eb9a9-86ca-488a-ac64-5a9393dc022f",
      "segmentation-type": "cisco-vts-identities:vlan",
      "segmentation-id": 1002,
      "mac-address": "bb:34:78:e6:44:9d"
     }
   ],
   "id": "b960a3e2-eb9f-43cf-98e7-090ad097d5cd",
   "name": "FIRSTTRUNK"
  }
}
```


GET
Show details of a VTS trunk with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/trunks/trunk/<id>

LIST (GET)
Show list of all VTS trunks
https://<vtc-ip>:8888/api/running/cisco-vts/trunks/trunk/

DELETE
Delete a VTS trunk with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/trunks/trunk/<id>

4.13 Router

Shows/Lists information for creates, updates and deletes of Routers

Create (PUT)
Create a router in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

Parameter	Value	Description
network-id	uuid	Network ID
id	uuid	Router ID
name	String	Router name
router-gateway	String	External network id, if exists
router-gateway-ip-address	String	Router gateway IPv4 address
router-gateway-ipv6-address	String	Router gateway IPv6 address

Sample Payload:


```
{
   "router": [
      {
```

```
            "id": "d8c58e0e-50ce-4ef8-923e-5d053ef38888",
            "status": "cisco-vts-identities:active",
            "name": "router3",
            "router-gateway": " dc44545b-0b95-4463-aad9-a310102e5f95 "
            "router-gateway-ip-address":null,
            "router-gateway-ipv6-address":null
        }
    ]
}
```


GET

Show router details associated to one VTS router under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

LIST (GET)

Show list of all routers under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/

DELETE

Delete router under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

4.14 Router Interfaces

Shows/Lists information for creates, updates and deletes of Interface

Create (PUT)

Create an interface in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

Parameter	Value	Description
Id	uuid	Interface ID
subnet-id	uuid	Subnet ID
router-id	uuid	Router ID

Sample Payload:


```
{
   "interface": [
```

```
        {
          "subnet-id": "3faf211d-1fca-4aff-9c1c-086cda772bae",
          "router-id": "3faf211d-1fca-4aff-9c1c-086cda777777",

        }
      ]
    }
```


GET

Show interface details associated to one VTS Interface under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

LIST (GET)

Show list of all interfaces under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/

DELETE

Delete interface under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

4.15 Infrastructure Policy

The following APIs are used to do CRUD operations on the infrastructure policy.

4.15.1 Administrative Domains

The following APIs are used to perform CRUD operations on administrative domains.

4.15.1.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains
https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/admin-domain/{name}

4.15.1.2 Payload fields

Field	Data Type	Description
Name	string	Name of the administrative domain. This field is required.

4.15.1.3 Payload example

```
{
  “name”: “my-admin-domain”
}
```

4.15.2 Layer 2 Gateway Groups

The following APIs are used to perform CRUD operations on layer 2 gateways groups.

4.15.2.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l2-gateway-groups/l2-gateway-group{name2}

4.15.2.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L2 gateway group. This field is required.

4.15.2.3 Payload example

```
{
 “name”: “my-L2”
}
```

4.15.3 Devices in a Layer 2 Gateway Groups

The following APIs are used to perform CRUD operations on devices belonging to a layer 2 gateways group.

4.15.3.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l2-gateway-groups/l2-gateway-group/{name2}/devices/device/{name3}

4.15.3.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L2 gateway. This field is required.

4.15.3.3 Payload example

```
{
 “name”: “my-l2-device”
}
```

4.15.4 Policy Parameters for a Layer 2 Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a layer 2 gateways group.

4.15.4.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}/ policy-parameters

4.15.4.2 Payload fields

Field	Data Type / Value	Description
distribution-mode	“decentralized-l2”	Mode of distribution
control-plane-protocol	“bgp-evpn” or “flood-and-learn”	Control plane protocol used to learn end point addresses.
arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	“multicast” or “ingress-replication”	Packet replication mechanism. It determines how packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.15.4.3 Payload example

```
{
  “distribution-mode”: “decentralized-l2”,
  “arp-suppression”: null
}
```

4.15.5 Parent of a Layer 2 Gateway Group

The following APIs are used to perform CRUD operations to assign a parent group to a layer 2 gateways group.

4.15.5.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}

4.15.5.2 Payload fields

Note that the ad-l3-gw-parent and l3-gw-parent are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS. l3-dc-gw-parent can be set in parallel to enable Layer 2 VNI service, extend the network to core.

Field	Data Type	Description

ad-l3-gw-parent	string	This L2 group's L3 gateway parent in the current administrative domain
l3-gw-parent	string	This L2 group's L3 gateway parent in the infrastructure policy hierarchy
l3-dc-gw-parent	string	This group's L3 datacenter gateway parent in the infrastructure policy hierarchy.

4.15.5.3 Payload example

```
{
  “ad-l3-gw-parent”: “my-l3-group”
}
```

or for L2 VNI service extension.

```
{
  “ad-l3-gw-parent”: “my-l3-group”,
  “l3-dc-gw-parent”: “my-dc-group”
}
```

4.15.6 Layer 3 Gateway Groups in an Administrative Domain

The following APIs are used to perform CRUD operations on layer 3 gateways groups in an administrative domain.

4.15.6.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l3-gateway-groups/l3-gateway-group{name2}

4.15.6.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L3 gateway group. This field is required.

4.15.6.3 Payload example

```
{
  “name”: “my-l3-group”
}
```

4.15.7 Devices in a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations on devices belonging to a layer 3 gateways group in an administrative domain.

4.15.7.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l3-gateway-groups/l3-gateway-group/{name2}/devices/device/{name3}

4.15.7.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L3 gateway in an administrative domain. This field is required.

4.15.7.3 Payload example

```
{
  “name”: “my-l3-device”
}
```

4.15.8 Policy Parameters for a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations on policy parameters for a layer 3 gateways group in an administrative domain.

4.15.8.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}/policy-parameters

4.15.8.2 Payload fields

Field	Data Type	Description
distribution-mode	“decentralized-l3” or “centralized-l3”	Mode of distribution
control-plane-protocol	“bgp-evpn” or “flood-and-learn”	Control plane protocol used to learn end point addresses.
arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	“multicast” or “ingress-replication”	Packet replication mechanism. It determines how

		packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.15.8.3 Payload example

```
{
  “packet-replication”: “multicast”
}
```

4.15.9 Parent of a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations to assign a parent group to a layer 3 gateways group in an administrative domain.

4.15.9.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}

4.15.9.2 Payload fields

Note that the fields below are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS.

Field	Data Type	Description
l3-dc-gw-parent	string	This L3 group's L3 datacenter gateway parent in the infrastructure policy hierarchy.
l3-dci-parent	string	This L3 group's datacenter gateway interconnect parent

4.15.9.3 Payload example

```
{
  “l3-dc-gw-parent”: “my-dc-group”
}
```

4.15.10 Layer 3 Gateway Groups

The following APIs are used to perform CRUD operations on layer 3 gateways groups.

4.15.10.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group{name}

4.15.10.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L3 gateway group. This field is required.

4.15.10.3 Payload example

```
{
  “name”: “my-other-l3-group”
}
```

4.15.11 Devices in a Layer 3 Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a layer 3 gateways group.

4.15.11.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/devices/device/{name2}

4.15.11.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L3 gateway in an administrative domain. This field is required.

4.15.11.3 Payload example

```
{
  “name”: “my-other-l3-device”
}
```

4.15.12 Policy Parameters for a Layer 3 Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a layer 3 gateways group.

4.15.12.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/policy-parameters

4.15.12.2 Payload fields

Field	Data Type	Description
distribution-mode	“decentralized-l3” or “centralized-l3”	Mode of distribution
control-plane-protocol	“bgp-evpn” or “flood-and-learn”	Control plane protocol used to learn end point addresses.

arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	“multicast” or “ingress-replication”	Packet replication mechanism. It determines how packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.15.12.3 Payload example

```
{
 “distribution-mode”: “decentralized-l2”,
 “arp-suppression”: null
}
```

4.15.13 Parent of a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations to assign a parent group to a layer 3 gateways group in an administrative domain.

4.15.13.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name2}

4.15.13.2 Payload fields

Note that the fields below are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS.

Field	Data Type	Description
l3-dc-gw-parent	string	This L3 group's L3 datacenter gateway parent in the infrastructure policy hierarchy.
l3-dci-parent	string	This L3 group's datacenter gateway interconnect parent

4.15.13.3 Payload example

```
{
 “l3-dci-parent”: “my-dci-group”
}
```

4.15.14 Data Center Gateway Groups

The following APIs are used to perform CRUD operations on data center gateway groups.

4.15.14.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group{name}

4.15.14.2 Payload fields

Field	Data Type	Description
Name	string	Name of the data center gateway group. This field is required.

4.15.14.3 Payload example

```
{
  “name”: “my-dc-group”
}
```

4.15.15 Devices in a Data Center Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a data center gateway group.

4.15.15.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/devices/device/{name2}

4.15.15.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as data center gateway This field is required.

4.15.15.3 Payload example

```
{
  “name”: “my-dc-device”
}
```

4.15.16 Policy Parameters for a Data Center Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a data center gateway group.

4.15.16.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/policy-parameters

4.15.16.2 Payload fields

Field	Data Type	Description
distribution-mode	“decentralized-l3” or “centralized-l3”	Mode of distribution

4.15.16.3 Payload example

```
{
  “distribution-mode”: “decentralized-l3”
}
```

4.15.17 Parent of a Data Center Gateway Group

The following APIs are used to perform CRUD operations to assign a parent group to a data center gateway group.

4.15.17.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}

4.15.17.2 Payload fields

Field	Data Type	Description
l3-dci-parent	string	This data center gateway group's datacenter gateway interconnect parent

4.15.17.3 Payload example

```
{
  “l3-dci-parent”: “my-dci-group”
}
```

4.15.18 Data Center Interconnect Groups

The following APIs are used to perform CRUD operations on data center interconect groups.

4.15.18.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group{name}

4.15.18.2 Payload fields

Field	Data Type	Description
Name	string	Name of the data center gateway group. This field is required.

4.15.18.3 Payload example

```
{
  “name”: “my-dci-group”
}
```

4.15.19 Devices in a Data Center Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a data center gateway group.

4.15.19.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group/{name}/devices/device/{name2}

4.15.19.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as data center gateway This field is required.

4.15.19.3 Payload example

```
{
 “name”: “my-dci-device”
}
```

4.15.20 Attach Stitching profile in a Data Center Interconnect group

The following APIs are used to perform CRUD operations on stitching profiles belonging to a data center interconnect group.

4.15.20.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dci-gateway-groups/l3-dci-group/{name}/ stitching-profiles/ stitching-profile/{id}

4.15.20.2 Payload fields

Field	Data Type	Description
Id	String	Reference to the stitching profile id.

4.15.20.3 Payload example

```
{
 “id”: “bc993e8c-6b4c-41c2-a4b4-46fc4dc1d3a2”
}
```

4.15.21 Provide redundancy group parameters in a Data Center Interconnect group

The following APIs are used to perform CRUD operations on redundancy group belonging to a data center interconnect group.

4.15.21.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dci-gateway-groups/l3-dci-group/{name}/redundancy-group

4.15.21.2 Payload fields

Field	Data Type	Description
Iccp group id	String	VXLAN (Fabric) Group Number Or MPLS (Core) Group Number Based on group type
Group-type	iccp-group-type	Possible values “iccp-fabric-group” or “iccp-core-group”
Esi – core-esi	String (9 octets)	Ethernet segment id for NVE overlay if configured.

4.15.21.3 Payload example

```
{
      "iccp":{
        "groups":{
          "group":[
            {
              "id":11,
              "group-type":"iccp-fabric-group"
            },
            {
              "id":12,
              "group-type":"iccp-core-group"
            }
          ]
        }
      },
      "esi":{
        "core-esi":"88.00.00.00.00.00.00.00.01"
      }
    }
```

4.16 Route Reflectors

4.16.1 Route Reflector Mode

The following APIs are used to perform CRUD operations on the route reflector mode.

4.16.1.1 URIs

https://host:8888/api/running/cisco-vts/global-settings

4.16.1.2 Payload fields

Field	Data Type	Description
route-reflector-mode	“global-rr” or “in-line-rr”	Determines how the route reflector operate

4.16.1.3 Payload example

```
{
  “route-reflector-mode”: “global-rr”;
}
```

4.16.2 Global Route Reflectors

The following APIs are used to perform CRUD operations on the set of global route reflectors.

4.16.2.1 URIs

https://host:8888/api/running/cisco-vts/global-settings/global-route-reflectors/global-route-reflector

https://host:8888/api/running/cisco-vts/global-settings/global-route-reflectors/global-route-reflector/{name}

4.16.2.2 Payload fields

Field	Data Type	Description
name	string	Name of the route reflector

4.16.2.3 Payload example

```
{
  “name”: “my-first-route-reflector”
}
```

4.17 Global Settings

The following APIs are used to do CRUD operations on global settings.

4.17.1 Anycast Gateway Address

The following APIs are used to perform CRUD operations on the common MAC address for all L2 Gateway devices on the Infrastructure Policy hierarchy.

4.17.1.1 URIs

https://host:8888/api/running/cisco-vts/global-settings/anycast-gateway/anycast-gw-address

4.17.1.2 Payload fields

Field	Data Type	Description
anycast-gw-address	MAC address e.g.: AA:BB:CC:DD:EE:FF	Common MAC address for all L2 Gateway devices

4.17.1.3 Payload example

```
{
  “anycast-gw-address”: “00:11:22:33:44:55”;
}
```

4.17.2 VTF Mode

The following APIs are used to perform CRUD operations on VTF mode to be used in the system.

4.17.2.1 URIs

https://host:8888/api/running/cisco-vts/global-settings/vtf-mode-config/vtf-mode

Field	Data Type	Description
vtf-mode	vtf-mode-enum values: l2, vtep	Global settings for the mode of operation of VTF mode.

4.18 Vcenter VTS Plugin API

Current VTS-VCenter plugin supports network provisioning operations through VCenter GUI extensions.

NOTE: The user has to log into vSphere WebClient and enter NCS credentials on the VTS plugin, before any of the rest calls are made.

Base URI:

https://<vCenterIP>:9443/eventMonitorDaemon/EventMonitorWebService

Headers

1.) Create/POST – Content-type – “application/json”
2.) Delete – Content-type – “text/plain”
3.) Authentication - (Key –Authorization, Value – (vc-username/password))

4.18.1 Resource: Network

Create

POST	/network/		Creates a new network

Sample Payload:

{"network": {"name": "TestNetwork1",

"admin-state-up": "true",

"provider-physical-network": "vlan",

"provider-network-type": "vts:vxlan",

"status": "vts:ACTIVE",

"router-external": "false",

"tenant-name": "Tenant1",

"dcname": "DatacenterSample",

"shared": "true"}}

Note:
"shared" is the attribute which indicates the network created is Shared Network or not. It is optional. Set it to “true” if you want the network to be a Shared Network. If “shared” attribute is not in the payload, by default it is set to “false”.

Lists

GET	/network/	Lists all the existing networks

Sample Payload:

{
"network" : [{"admin-state-up":"true",
"name":"net8",
"provider-network-type":"vts:vxlan",
"provider-physical-network":"vlan",
"router-external":"false",
"status":"vts:ACTIVE",
"tenant-id":"tenant1",
"tenant-name":"tenant1"
},
{"admin-state-up":"true",
"name":"net9",
"provider-network-type":"vts:vxlan",

"provider-physical-network":"vlan",
"router-external":"false",
"status":"vts:ACTIVE",
"tenant-id":"tenant1",
"tenant-name":"tenant1",
}]
}

Delete

DELETE	/network/{network_name}?dcname={dcname}	Deletes an existing network given its name and data center name on which it resides.

Show

GET	/network/{network_name}/	Lists details of given network named {network_name}

Sample Payload:

{"network":
{"admin-state-up":"true",
"name":"net8",
"provider-network-type":"vts:vxlan",
"provider-physical-network":"vlan",
"router-external":"false",
"status":"vts:ACTIVE",
"tenant-id":"D32508B0-40FC-27A2-B78F-462A08AB44A3",
"tenant-name":"tenant1",
}}

4.18.2 Resource: Subnet

Create

POST	/subnet/	Creates a new subnet

Sample Payload:

{
"subnet": [{
"cidr": "29.0.0.0/24",
"enable-dhcp": "false",
"gateway-ip": "29.0.0.1",
"ip-version": "4",
"name": "Subnet23",
"network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
"shared": "false",

"tenant-id": "tenant1",
"tenant-name": "tenant1"
}]
}

Lists

GET	/subnet	Lists all the existing subnets

Sample Payload:

{
"subnet": [
{
"cidr": "19.0.0.0/24",
"enable-dhcp": "false",
"gateway-ip": "19.0.0.1",
"id": "58717685-c9d1-4aeb-918e-3e0f37034ab5",
"ip-version": "4",
"name": "Subnet13",
"network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
"shared": "false",
"tenant-id": "tenant1",
"tenant-name": "tenant1"
},
{
"cidr": "29.0.0.0/24",
"enable-dhcp": "false",
"gateway-ip": "29.0.0.1",
"id": "c626fc93-0708-45eb-b578-0edee528736d",
"ip-version": "4",
"name": "Subnet23",
"network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
"shared": "false",
"tenant-id": "tenant1",
"tenant-name": "tenant1"
}
]
}

Delete

DELETE	/subnet/{subnet_name}/	Deletes a subnet named {subnet_name}

Show

GET	/subnet/{subnet_name}/	Lists details of given subnet named {subnet_name}

Sample Payload:

```
{
   "subnet": {
      "cidr": "29.0.0.0/24",
      "enable-dhcp": "false",
      "gateway-ip": "29.0.0.1",
      "id": "c626fc93-0708-45eb-b578-0edee528736d",
      "ip-version": "4",
      "name": "Subnet23",
      "network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
      "shared": "false",
      "tenant-id": "tenant1",
      "tenant-name": "tenant1"
   }
}
```


4.18.3 Resource: Router

Create

POST	/router	Creates a new router

Sample Payload:


```
{ "router" : [{
   "status" : "ACTIVE",
   "name" : "Router100",
   "tenant-id" : "Tenant1",
   "vni_number" : 5170,
   "router-gateway" : null
}]
}
```


Lists

GET	/router	Lists all the existing routers

Sample Payload:


```
{"router": [
      {
         "id": "4b4bb642-3238-4d31-acb0-56e90a19cf01",
         "name": "Router2",
         "status": "vts:ACTIVE",
         "tenant-id": "tenant1",
         "vni_number": "12007"
      },
      {
         "id": "4d602cb8-4f93-43d1-87e0-fd943f18642e",
         "name": "Router5",
         "status": "vts:ACTIVE",
         "tenant-id": "tenant1",
         "vni_number": "12002"
      }
```

```
   ]
}
```

Delete

DELETE	/router/{router_name}/	Deletes a router named {router_name}

Show

GET	/router/{router_name}/	Lists details of given router named {router_name}

Sample Payload:

```
{
   "router": {
      "id": "4d602cb8-4f93-43d1-87e0-fd943f18642e",
      "name": "Router5",
      "status": "vts:ACTIVE",
      "tenant-id": "tenant1",
      "vni_number": "12002"
   }
}
```

4.18.4 Resource: Interface

Create

POST	/interface	Attach Interface to the router

Sample Payload:

```
{"interfaces" : [{
   "subnet-id": "aabc0bbf-1c63-46a7-99b6-8c7fa91a9ae6",
   "router-id": "aac12354-3573-4472-9370-5e2726bc27eb",
   "ip-address": "46.0.0.1",
   "port-create": "l3:none"
   }]
}
```

Lists

GET	/interface	Lists all the existing interfaces

Sample Payload:

```
{
    "interfaces": [
   {
   "subnet-id": "114d1323-2458-413f-860d-20bed461fc0b",
   "router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
   "ip-address": "29.0.0.1",
   "port-create": "l3:none"
```

```
   },
   {
   "subnet-id": "909953fc-9b5c-495d-b428-5464d969101b",
   "router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
   "ip-address": "14.0.1.1",
         "port-create": "l3:none"
    }
  ]
 }
```


Show

GET	/interface/{subnet_name}/	Lists details of given interface with subnet name as {subnet_name }

Sample Payload:


```
{“interfaces” : [{
     "subnet-id": "114d1323-2458-413f-860d-20bed461fc0b",
     "router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
     "ip-address": "29.0.0.1",
     "port-create": "l3:none"
     }]
}
```


Delete

DELETE	/interface/{subnet_name}/	Detaches subnet named {subnet_name} from router

External Network Attach

POST	/interface/{router_name}/{external_name}?action=attach	Attaches external network named {external_name} to router {router_name}

External Network Detach

POST	/interface/{router_name}/{external_name}?action=detach	Detaches external network named {external_name} to router {router_name}

4.19 Route Template API

4.19.1 Create Route Template

POST	/templates	Create a new configuration template in VTS

Sample Payload:


```
{
  "template":[
    {
      "name":"Sample Template",
      "description":"A sample router template",
      "type":"route",
      "created-on":"2016-01-12T09:05:00",
      "modified-on":"2016-01-12T09:30:00",
      "route-template":{
        "rt-seed":"70000",
        "vrf-name":"Sample VRF name",
        "static-routes":{
          "static-route":[
            {
              "destination":"1.1.1.0/24",
              "next-hop":"2.2.2.0/24"
            },
            {
              "destination":"3.3.3.0/24",
              "next-hop":"4.4.4.4/32"
            }
          ]
        },
        "import-route-targets":{
          "import-route-target":[
            {
              "stitching":"",
              "route-target":"100:1000",
              "route-target-type":"route-target-internal"
            },
            {
              "route-target":"192.168.1.2:3001",
              "route-target-type":"route-target-external"
            }
```

```
        ]
      },
      "export-route-targets":{
        "export-route-target":[
          {
            "route-target":"172.27.184.56:12345",
            "route-target-type":"route-target-internal"
          },
          {
            "stitching":"",
            "route-target":"123:60000",
            "route-target-type":"route-target-external"
          }
        ]
      }
    }
  ]
}
```


Fields:

Field Name	Possible Value	Mandatory
name	A string with a maximum of 128 characters	Y
description	A string with a maximum of 128 characters	N
type	In this release, there is only one template type: route	Y
created-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
modified-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
vrf-name	A string with a maximum of 128 characters	N
rt-seed	An integer value with a range of 1-16777215	N
destination	An IPv4 address with prefix length	Y (if a new static-route element is to be added)

next-hop	An IPv4 address (Note: the prefix length information is optional in this case)	Y (if a new static-route element is to be added)
stitching	Can be either present or not present	N
route-target	Can be of one of the following 4 formats: 1. ASN2:NN4 2. ASN4:NN2 3. IPv4:NN2 where: NN2 and ASN2 has a range of 1-65535 NN4 and ASN4 has a range of 1-4294967295 IPv4 is an IPv4 address in the dotted decimal format	Y (if a new import-route-target/export-route-target element is to be added)
route-target-type	Can be of one of the following types: route-target-internal, route-target-external, route-target-both	Y (if a new import-route-target/export-route-target element is to be added)

4.19.2 Updating a Template

*Note: In this case, only the fields specified in the JSON body will be updated. All the existing fields will remain the same.

PATCH	/templates/template/<template_name>	Update a template

Sample Payload:

```
{
  "template":[
    {
      "router-template":{
        "rt-seed":"8000",
        "static-routes":{
          "static-route":[
            {
              "destination":"10.10.10.0/24",
              "next-hop":"20.20.20.0/24"
            }
          ]
        },
        "import-route-targets":{
          "import-route-target":[
```

```
            {
              "stitching":"",
              "route-target":"8888:99",
              "route-target-type":"route-target-internal"
            }
          ]
        }
      }
    }
  ]
}
```


4.19.3 Get List of Templates

GET	/templates	Retrieve list of templates

Sample Payload:


```
{
  "cisco-vts-templates:templates":{
    "template":[
      {
        "name":"SR_RT_Template_1"
      },
      {
        "name":"SR_Template_1"
      },
      {
        "name":"Sample Template"
      },
      {
        "name":"Temp_3"
      },
      {
        "name":"Temp_4"
      }
    ]
  }
}
```

4.19.4 Get a template using name

GET	/templates/template/{template_name>}	Gets details about a particular template

Sample Payload:


```
{
  "cisco-vts-templates:template":{
    "name":"TemplateA",
    "description":"TempA",
    "type":"cisco-vts-templates:route",
    "created-on":"2016-02-19T23:30:12.128+00:00",
    "modified-on":"2016-02-19T23:30:12.129+00:00",
    "route-template":{
      "static-routes":{
        "static-route":[
          {
            "destination":"1.1.1.0/24",
            "next-hop":"2.2.2.0/24"
          }
        ]
      },
      "import-route-targets":{
        "import-route-target":[
          {
            "route-target":"2.2.2.0:24"
          }
        ]
      },
      "rt-seed":12345
    },
}
```


4.19.5 Delete all templates

DELETE	/templates	Deletes all templates. Note: Template can be deleted if its not attached to Tenant or Router.

4.19.6 Delete a template identified by name

DELETE	/templates/template/{template-name}	Deletes a template identified by name. Note: Template can be deleted if its not attached to Tenant or Router.

4.19.7 Get Route Template attached to Tenant

GET	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map	Lists existing tenant assigned to tenant

Sample Payload:


```
{
  "collection":{
    "cisco-vts:template-map":[
      {
        "template-type":"route",
        "template-name":"TemplateB"
      }
    ]
  }
}
```


4.19.8 Attach Route Template to Tenant

PUT	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map	Attaches a pre-defined template to a tenant

Sample Payload:


```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```


4.19.9 Detach Route Template From Tenant

DELETE	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map/route	Detach assigned template from tenant

4.19.10 Get route template attached to Router

GET	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/{topology id}/routers/router/{router id}/template-maps/template-map	Lists existing template assigned to Router

Sample Payload:


```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```


4.19.11 Attach Route Template to Router

PUT	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/topology id}/routers/router/{router id}/template-maps/template-map/route	Attaches a pre-defined template to a router

Sample Payload:


```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```


4.19.12 Detach Route Template From Router

DELETE	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/{topology id}/routers/router/{router id}/template-maps/template-map/route	Detach Assigned Template From Router

4.19.13 Get Template Type

GET	/templates/template/{template name}/type	For a given template, list the type

Sample Payload:

```
{
  "type": "route"
}
```

4.19.14 Get List of Import Route Targets

GET	/templates/template/{template-name}/route-template/import-route-targets/import-route-target	For a given template, get list of import route targets

Sample Payload:

```
{
  "import-route-target":[
    {
      "route-target":"65001:23",
      "route-target-type":"route-target-internal"
    }
  ]
}
```

4.19.15 Get List of Export Route Targets

GET	/templates/template/{template-name}/route-template/export-route-targets/export-route-target	For a given template, get list of export route targets

Sample Payload:

```
{
  "export-route-target":[
    {
      "route-target":"65004:22",
      "route-target-type":"route-target-internal"
    }
  ]
}
```

4.19.16 Set Import Route Target to Route Template

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target}	For a given template, add a import route target

Sample Payload:

```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"cisco-vts-templates:route-target-external"
    }
  ]
}
```


4.19.17 Set Export Route Target to Route Template

PUT	/templates/template/{template-name}/route-template/export-route-targets/export-route-target/65004:23	For a given template, add an export route target

Sample Payload:


```
{
  "export-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"cisco-vts-templates:route-target-external"
    }
  ]
}
```


4.19.18 Set Route Target to Internal Device

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target}	For a given template, add a route target to a ToR

Sample Payload:


```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"cisco-vts-templates:route-target-internal"
    }
  ]
}
```


4.19.19 Set Route Target to External Device

PUT	/templates/template/{template-name}/route-template/import-	For a given template, add a route target to a DCI

	route-targets/import-route-target/{route target}	

Sample Payload:


```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"cisco-vts-templates:route-target-external"
    }
  ]
}
```


4.19.20 Set Route Target to Both Internal and External Device

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target}	For a given template, add a route target to both a ToR and DCI

Sample Payload:


```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"cisco-vts-templates:route-target-both"
    }
  ]
}
```


4.19.21 Get List of Static Routes in Route Template

GET	/templates/template/{template-name}/route-template/static-routes	For a given template, get list of export route targets

Sample Payload:


```
{
  "cisco-vts-templates:static-routes":{
    "static-route":[
      {
        "destination":"1.1.1.0/24",
        "next-hop":"2.2.2.0/24"
      },
      {
        "destination":"10.10.10.0/24",
        "next-hop":"20.20.20.0/24"
```

```
      }
    ]
  }
}
```

4.19.22 Set a static route to a route template

PATCH	/templates/template/{template-name}/route-template/static-routes/static-route/	For a given template, add a static route

Sample Payload:

```
{
    "static-route":[
      {
        "destination":"8.8.8.8/32",
        "next-hop":"4.4.4.4"
      }
    ]
}
```

4.19.23 Delete a specific static route from Route Template

DELETE	/templates/template/{template-name}/route-template/static-routes/static-route/”<destination_to_be_deleted>,<next_hop_to_be_deleted>”	Delete a specific static route in an existing template

4.19.24 Delete a specific import route target from route template

DELETE	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/"<route_target_to_be_deleted>"	Delete a specific import route target in an existing template

4.19.25 Delete a specific export route target from route template

DELETE	/templates/template/{template-name}/route-template/export-route-targets/export-route-target/"<route_target_to_be_deleted>"	Delete a specific export route target in an existing template

4.20 L3 Service Extension Template API

4.20.1 Create a L3 service extension template

POST	/templates	Create a new L3 service extension template in VTS

Sample Payload:


```
{
  "template":[
    {
      "name":"Sample L3 Template",
      "description":"A sample L3 service extension template",
      "type":"cisco-vts-templates:l3-service-extension",
      "created-on":"2016-05-31T09:05:00",
      "modified-on":"2016-05-31T09:30:00",
      "l3-service-extension-template":{
        "schema-revision-date":"2016-05-31",
        "schema-namespace":"http://tail-f.com/ned/cisco-nx",
        "keypath-values":{
          "keypath-value":[
            {
              "keypath":"config/nx:router/bgp{100}/vrf",
              "value":"{\"vrf\":{\"name\":\"admin-test\"}}"
            },
            {
              "keypath ":"config/nx:vrf/context{admin-TestRouter}/ip",
              "value ":"{\"ip\":{\"domain-name\":\"TestDomain\"}}"
            }
          ]
        }
      }
    }
```

```
        ]
    }
```

Fields:

Field Name	Possible Value	Mandatory
name	A non-empty string with a maximum of 128 characters	Y
description	A string with a maximum of 128 characters	N
type	The type for L3 service extension template is ‘‘cisco-vts-templates:l3-service-extension’’	Y
created-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
modified-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
schema-revision-date	String in YYYY-MM-DD format indicating the schema yang's revision date	Y (if the l3-service-extension template container is specified)
schema-namespace	Namespace for the service extension schema	Y (if the l3-service-extension template container is specified)
keypath	A unique path to the service extension config	Y (if a keypath is to be added)
value	The config value (in valid JSON format) for the corresponding key path	Y (if a keypath is specified)

4.20.2 Updating a L3 service extension template

*Note: When updating a template, the "modified-on" field is a REQUIRED field in the payload. This is to avoid the situation where a user is updating an older version of a template when a newer version is available. A user can get the "modified-on" value upon retrieving a template.

If the "modified-on" value in the latest version of the template does not match with the "modified-on" value sent in the payload of the template update request, the update request will be REJECTED, in which case the user will need to fetch the latest "modified-on" value again by retrieving the latest version of the template.

PUT	/templates/template/<template_name>	Update a L3 service extension template

Sample Payload:


```
{
  "template":[
    {
      "name":"Sample L3 Template",
      "description":"Updated description",
      "type":"cisco-vts-templates:l3-service-extension",
      "created-on":"2016-05-31T09:05:00",
      "modified-on":"2016-05-31T09:30:00",
      "l3-service-extension-template":{
        "schema-revision-date":"2016-05-31",
        "schema-namespace":"http://tail-f.com/ned/cisco-nx",
        "keypath-values":{
          "keypath-value":[
            {
              "keypath":"config/nx:router/bgp{100}/vrf",
              "value":"{\"vrf\":{\"name\":\"updated-vrf-name\"}}"
            },
            {
              "keypath ":"config/nx:vrf/context{admin-TestRouter}/ip",
              "value ":"{\"ip\":{\"domain-name\":\"updated-domain-name\"}}"
            }
          ]
        }
      }
    }
  ]
}
```


4.20.3 Get List of Templates

*Note: This API will get a list of all templates in the system, including both Route and L3 Service Extension templates.

GET	/templates	Retrieve list of templates

Sample Payload:


```
{
  "cisco-vts-templates:templates":{
    "template":[
      {
        "name":"Sample_Route_Template"
      },
      {
        "name":"Sample L3 Template"
      },
      {
        "name":"Another_L3_Template"
      }
    ]
  }
}
```


4.20.4 Get a specific template using name

GET	/templates/template/<template_name>	Gets details about a particular template

Sample Payload:


```
{
  "cisco-vts-templates:template":{
    "name":"Sample L3 Template",
    "description":"A sample L3 service extension template",
    "type":"cisco-vts-templates:l3-service-extension ",
    "created-on":"2016-05-31T09:05:00",
    "modified-on":"2016-05-31T09:30:00",
    "l3-service-extension-template":{
      "schema-revision-date":"2016-05-31",
      "schema-namespace":"http://tail-f.com/ned/cisco-nx",
      "keypath-values":{
        "keypath-value":[
          {
            "keypath":"config/nx:router/bgp{100}/vrf",
            "value":"{\"vrf\":{\"name\":\"updated-vrf-name\"}}"
          },
          {
            "keypath":"config/nx:vrf/context{admin-TestRouter}/ip",
            "value":"{\"ip\":{\"domain-name\":\"updated-domain-name\"}}"
          }
        ]
      }
    }
}
```

4.20.5 Delete all templates

*Note: This API will delete all templates in the system, including both Route and L3 Service Extension templates.

DELETE	/templates	Deletes all templates. Note: A template can only be deleted if it is not currently attached to a Tenant or Router.

4.20.6 Delete a template identified by name

DELETE	/templates/template/<template-name>	Deletes a template identified by name. Note: A template can only be deleted if it is not currently attached to a Tenant or Router.

4.20.7 Create a template device group

*Note: Currently VTS does not support the modification of the device list using PUT/PATCH API once the template device group is created. To add or remove devices from the template device group, a user can first use the DELETE API to delete the existing template device group, and then use the POST API to create a new template device group with the updated device list.

POST	/cisco-vts/template-device-groups	Create a template device group

Sample Payload:

```
{
  "template-device-group":[
    {
      "device-group-name":"system-device-group-1",
      "devices":{
       "device":[
         {
           "name":"vts-142-tor1"
         }
       ]
      }
    }
```

```
    ]
}
```

Fields:

Field Name	Possible Value	Mandatory
device-group-name	A non-empty string with a maximum of 128 characters. A device group name of ‘‘ALL’’ is used to indicate that the device group contains all devices, in which case the name field will have a value of ‘‘*’’.	Y
name	The name of the device that belongs to this device group. In the case where the device-group-name is ‘‘ALL’’, this field will have a value of ‘‘*’’.	Y

4.20.8 Get a list of all template device groups

GET	/cisco-vts/template-device-groups	Get a list of all the template device groups

Sample Payload:


```
{
  "template-device-groups":{
    "template-device-group":[
      {
        "device-group-name":"system-device-group-1",
        "devices":{
          "device":[
            {
              "name":"vts-142-tor1"
            }
          ]
        }
      },
      {
        "device-group-name":"system-device-group-2",
        "devices":{
          "device":[
            {
              "name":"vts-144-tor2"
            }
```

```
        ]
      }
    }
  ]
 }
}
```

4.20.9 Get a specific template device group

GET	/cisco-vts/template-device-groups/template-device-group/<name_of_device_group>	Get a specific template device group

Sample Payload:

```
{
  "template-device-group":[
    {
      "device-group-name":"system-device-group-1",
      "devices":{
       "device":[
        {
          "name":"vts-142-tor1"
        }
       ]
      }
    }
  ]
}
```

4.20.10 Delete a specific template device group

DELETE	/cisco-vts/template-device-groups/template-device-group/<name_of_device_group_to_be_deleted>	Delete a given template device group

4.20.11 Attach L3 service extension template to a tenant or router

POST	/cisco-vts/template-target-maps	Attaches an existing l3 service extension template to a tenant/router

Sample Payload:

```
{
  "template-target-map":[
    {
      "template-target-id":"/vts:cisco-vts/tenants/tenant{admin}",
```

```
      "template-target-type":"tenant-target",
      "template-type":"cisco-vts-templates:l3-service-extension",
      "template-name":"Sample L3 Template",
      "device-group-id":"system-device-group-1",
      "last-changed":"2014-05-12T12:00:00-00:00",
    }
  ]
}
```


Fields:

Field Name	Possible Value	Mandatory
template-target-id	The full path to the tenant (in the case of tenant association) or the full path to the router (in the case of router association).	Y
template-target-type	The possible template target types are: vts:tenant-target (in the case of tenant association) or vts:router-target (in the case of router association)	N
template-type	The template type for L3 service extension template is ‘‘cisco-vts-templates:l3-service-extension’’	Y
template-name	The name of the template to be associated.	Y
device-group-id	The name of the device group indicating the group of devices to be applied for this template-target association.	N
last-changed	A timestamp indicating when template configuration applied on devices has changed.	N

4.20.12 Get a list of all the existing L3 service extension template associations

GET	/cisco-vts/template-target-maps	Gets a list of all the L3 service extension templates that are currently associated with a tenant/router

Sample Payload:


```
{
  "template-target-maps":{
    "template-target-map":[
      {
        "template-target-id":"/vts:cisco-vts/tenants/tenant{admin}",
        "template-target-type":"tenant-target",
```

```
        "template-type":"cisco-vts-templates:l3-service-extension",
        "template-name":"Sample L3 Template",
        "device-group-id":"system-device-group-1",
        "last-changed":"2014-05-12T12:00:00-00:00",
      },
      {
        "template-target-id":"/vts:cisco-
vts/tenants/tenant{admin}/toplogies/topology{admin}/routers/router{E24164F7-
EC4E-4459-A533-301768C5C01E}",
        "template-target-type":"router-target",
        "template-type":"cisco-vts-templates:l3-service-extension",
        "template-name":"Another L3 Template",
        "device-group-id":"system-device-group-2",
        "last-changed":"2014-05-12T12:00:00-00:00",
      }
    ]
  }
}
```


4.20.13 Detach L3 service extension template from tenant or router

DELETE	/cisco-vts/template-target-maps/"<template_target_id>,<template_target_type>,<template_type>"	Detach a given l3 service extension template from specified tenant/router

4.21 Resource Pools

4.21.1 Vni Pool Creation

PUT	/resource-pools/vni-pool/vnipool	Create the global vni pool. Pool name must be "vnipool"

Sample Payload:


```
{
  "vni-pool": {
    "name":"vnipool",
    "ranges":{
      "range":{
        "id":"78d07cfe-b944-49a4-a598-3bcf0b843625",
        "start":"6001",
```

```
      "end":"7000"
    }
   }
  }
}
```

4.21.2 Vlan Pool Creation

PUT	/resource-pools/vlan-pool/<device name>	Create a vlan pool for a given device

Sample Payload:

```
{
  "vlan-pool": {
    "name":"n9k",
    "ranges":{
      "range":{
        "id":"78d07cfe-b944-49a4-a598-3bcf0b843625",
        "start":"1001",
        "end":"2000"
      }
    }
  }
}
```

4.21.3 Interface Dot1Q Pool Creation

PUT	/resource-pools/ device-interface-pool/<device name>	Create an Interface Dot1Q pool for a given device (Only Nexus 7000 devices)

Sample Payload:

```
{
  "device-interface-pool": {
    "name":"n7k",
    "ranges":{
      "range":{
        "id":"78d07cfe-b944-49a4-a598-3bcf0b843625",
        "start":"1001",
        "end":"2000"
      }
    }
  }
}
```

4.21.4 Multicast Pool Creation

PUT	/resource-pools/vni-pool/multicastpool	Create the global multicast pool. Pool name must be “multicastpool”

Sample Payload:

```
{
  "multicast-pool": {
    "name":"multicastpool",
    "ranges":{
      "range":{
        "id":"78d07cfe-b944-49a4-a598-3bcf0b843625",
        "start":"239.0.0.0",
        "end":"239.0.0.255"
      }
    }
  }
}
```

4.21.5 Create a Restricted Range

PUT	/resource-pools/vni-pool/vnipool	Create a range in a pool to be restricted

Sample Payload:

```
{
  "vni-pool": {
    "name":"vnipool",
    "ranges":{
      "range":{
        "id":"78d07cfe-b944-49a4-a598-3bcf0b843625",
        "start":"6001",
        "end":"7000"
        "restricted":""
      }
    }
  }
}
```

4.21.6 Unrestricting a Range

DELETE	/resource-pools/vni-pool/vnipool/ranges/range/<range id>/restricted	Removing the restrict attribute of a range

4.21.7 Restricting static allocations to range only

PUT	/cisco-vts/global-settings/resource-allocator-config/allocate-outside-range	Force static allocations to come only from ranges

Sample Payload:
{ "allocate-outside-range":"false"}

4.22 Profiles

Allows to manipulate profiles in VTS. Profiles is essentially a logical group of common attributes that can be attached to an existing VTS entities. For example, currently, “Stitching Profile” can be created and attached to DCI functional group in admin domain. This allows to provision some extra services on DCI devices, like L2 VNI.

4.22.1 Stitching profile

4.22.1.1 Create/Update URI

PATCH	/profiles/profile/	Create Stitching profile

4.22.1.2 Sample Payload

```
{
  "profile":{
    "id":"44d738a2-d901-11e6-9c75-005056862217",
    "name":"automationStitchingProfile",
    "type":"stitching",
    "created-on":"2017-01-12T11:56:55",
    "control-protocol-type":"cisco-vts-stitching-profile:control-protocol-mp-bgp",
    "vts-mp-bgp-profile-id":"4460c0a0-d901-11e6-9c75-005056862217",
    "services":{
      "service":[
        {
          "service-type":"cisco-vts-stitching-profile:service-type-internet",
          "route-policy-profile-id":"44a09ebe-d901-11e6-9c75-005056862217"
        }
      ]
     },
    "remote-neighbours":{
      "remote-neighbour":[
        {
          "router-id":"60.60.60.1"
        }
      ]
    }
  }
}
```

4.22.1.3 Payload Fields

Field	Data Type	Mandatory	Description
id	String	Y	Unique identifier
name	String	Y	Name of the profile

description	String	N	Details description if any
type	cisco-vts-profiles:profile-type	Y	This has to cisco-vts-profiles:stitching for stitching profile
remote-neighbours	List of neighbor ips	Y	At least 1 neighbor is needed
services	List	Y	List of services(intern/mpls l2vni)
service-type	cisco-vts-stitching-profile: core-network-service-type	Y	supported values are cisco-vts-stitching-profile:service-type-mpls-l2vpn and cisco-vts-stitching-profile:service-type-internet
control-protocol-typ	cisco-vts-stitching-profile: core-network-control-protocol	Y	Only supported value so far is cisco-vts-stitching-profile:control-protocol-mp-bgp
vts-mp-bgp-profile-id	Reference of BGP profile Id	Y	stitching

4.22.2 BGP profile

4.22.2.1 Create/Update URI

PATCH	/profiles/profile/	Create BGP profile

4.22.2.2 Sample Payload

```
{
  "profile":{
    "name":"bgp-profile",
    "type":"cisco-vts-profiles:bgp",
    "id":"c144f5ef-2cce-4b09-b496-987ff820de9d",
    "bgp-as-number":100,
    "address-families":[
      {
        "address-family":"l2vpn"
      }
```

```
    ],
    "loopback-if-number":1
  }
}
```

4.22.2.3 Payload Fields

Field	Data Type	Mandatory	Description
name	String	Y	Name of the profile
description	String	N	Details description if any
type	cisco-vts-profiles:profile-type	Y	This has to cisco-vts-profiles:bgp for bgp profile
bgp-as-number	String	Y	ASN number for core network
Address-families	List of families supported	Y	Only supported value so far l2vpn

4.22.3 Route policy profile

4.22.3.1 Create/Update URI

PATCH	/profiles/profile/	Create route policy profile

4.22.3.2 Sample Payload

```
{
  "profile":{
    "name":"router-profile",
    "type":"cisco-vts-profiles:route-policy",
    "id":"48763f5e-3444-4809-a491-dddbcbf28963",
underlay-route-policy":{
    "dc-fabric-external-policy":{
      "route-advertisements":{

      },
      "route-filter":"route-policy-in"
    },
    "dc-fabric-internal-policy":{
      "route-filter":"route-policy-out"
    }
  }
}
}
```

4.22.3.3 Payload Fields

Field	Data Type	Mandatory	Description
name	String	Y	Name of the profile
description	String	N	Details description if any
type	cisco-vts-profiles:profile-type	Y	This has to cisco-vts-profiles:route-policy for route policy profile
dc-fabric-external-policy	String	Y	Name of policy to be applied on DCI devices.
dc-fabric-internal-policy	String	Y	Name of policy to be applied on Fabric side devices.

4.22.4 Get any profile

GET	/profiles/profile/{id}	Get any profile

4.22.5 Delete any profile

Delete	/profiles/profile/{id}	Get any profile

4.23 Global Route Table (GRT) service profile (Route Leak profile)

Global route leaking feature enables you to provide internet/external connectivity to the host inside the Data Center. This feature allows associating/dissociating of Global Route Leaking (also known as Global Routing Table [GRT]) Service to/from the Overlay Router. The following should be done in order to utilize the GRT service. Pre-requisite for GRT is to create a Stitching profile with cisco-vts-stitching-profile:service-type-internet (refer 4.20.1) , BGP Profile (refer 4.20.2)

4.23.1 Create/Update a GRT Profile URI

PATCH	/profiles/profile/	Create overlay route policy profile

4.23.2 Sample payload

```
{
 "profile": {
  "id" : "39ca68b7-c522-4d7e-89e4-3019fc526333",
            "name": "Tenant100Grt100",
            "type": "cisco-vts-profiles:route-policy-overlay",
            "overlay-route-policy": {
               "dc-fabric-external-policy": {
                  "route-filter": "data-center-vrf-export-policy"
               },
               "dc-fabric-internal-policy": {
                  "route-filter": "data-center-vrf-import-policy"
               }
            }
         }
}
```

Field	Data Type	Mandatory	Description
name	String	Y	Name of the profile
description	String	N	Details description if any
type	cisco-vts-profiles:profile-type	Y	This has to be cisco-vts-profiles:route-policy-overlay for GRT service profile
dc-fabric-external-policy	String	Y	Name of already created route-policy filter on DCI -- to be applied as fabric export policy
dc-fabric-internal-policy	String	Y	Name of already created route-policy filter on DCI, to be applied as Fabric import policy to the overlay router.

4.23.3 GET VPN for Overlay Route Policy Profile

Fetch VPN id for overlay route-policy profile created in the step above.

GET	/vpns/vpn {id}	Get all VPN

RESULT:


```
{
 "cisco-vts:vpn": {
  "id": "20584262-7568-4068-8b09-3c0b7b0106b4",
  "name": "Tenant100GrtVpn1",
  "type": "cisco-vts:vpn-service-type-internet",
  "service-domain": "cisco-vts:vpn-service-domain-dc-fabric-external",
  "internet": {
   "address-families": {
    "address-family": [
     {
      "family": "ipv4",
      "overlay-route-policy": "39ca68b7-c522-4d7e-89e4-3019fc526333"
     }
    ]
   }
  }
 }
}
```


4.23.4 Associate VPN ID to Router

PAT-CH	tenants/tenant/Tenant100/topologies/topology/Tenant100/routers/router/	Associate VPN ID with Router


```
{
 "cisco-vts:router": {
  "id": "d640096b-1a42-40b6-88f5-7ca8050bb511",
  "name": "T100RT1",
  "router-gateway": "d8451c2d-5a56-4bcd-bbbd-cf78202017c4",
  "status": "cisco-vts-identities:active",
  "vts-allocated-vni": 10231,
  "overlay-router-vpn-id": "20584262-7568-4068-8b09-3c0b7b0106b4"
 }
}
```


Overlay-router-vpn-id	String	VPN id associate with overlay route policy profile is creation

4.24 AAA

Allows to manipulate AAA server and accounting configuration in VTS.

4.24.1 Authentication and Authorization

4.24.1.1 Global Timeout – Get and Update

Global timeout is the connection timeout of VTC with a AAA server. The default is 15 secs.

GET	/cisco-vts/external-authentication/timeout	Get the Global Timeout

Sample Response

```
{
  "cisco-vts:timeout": 15

}
```

PATCH	/cisco-vts/external-authentication/timeout	Set the Global Timeout

Sample Payload

```
{

"cisco-vts:timeout": 20

}
```

4.24.1.2 Enable and Disable External Auth

In order to use AAA, external auth needs to be enabled.

PATCH	/cisco-vts/external-authentication	Enable External auth by setting the external-auth flag

Sample Payload

```
{
"external-authentication": {
        "external-auth-enabled-info": {
                "external-auth-enabled": [null]
    }
}
}
```


DELETE	/cisco-vts/external-authentication/external-auth-enabled-info/external-auth-enabled	Disable External auth by deleting the external-auth flag

4.24.1.3 Tacacs+ Server Configuration – Create, Update, Delete and Get

A Tacacs+ server can be configured in VTC by adding a Tacacs+ server configuration. This will enable VTC to communicate with Tacacs+ server for authentication, authorization and accounting.

PATCH	/cisco-vts/external-authentication/external-auth-modes/external-auth-mode	Create an Tacacs External Auth configuration in VTC

Sample Payload


```
{
 "external-auth-mode" :  {
   "order": 0,
    "auth-type": "tacacs-external-auth-mode",
    "auth-mode-enabled": [null],
    "tacacs-server-credentials": {
        "tacacs-server-credential": [
            {
               "key": "tacacskey123",
               "port": "49”,
                "ip-hostname": "172.20.100.205",
```

```
                "id": "48763f5e-3444-4809-a491-dddbcbf28963"
              }
            ]
        }
    }
}
```


GET	/cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/<tacacs-server-uuid>	Get the details of the a tacacs server credential

Sample Response

GET /cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/0ca6434c-408e-42cb-ac59-7acdca56d798


```
{
  "cisco-vts:tacacs-server-credential": {
    "id": "0ca6434c-408e-42cb-ac59-7acdca56d798",
    "ip-hostname": "172.20.100.200",
    "key": "cisco",
    "port": 49
  }
}
```


DELETE	/cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/<tacacs-server-uuid>	Delete a tacacs server configured using the uuid

Sample Request

DELETE /cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/0ca6434c-408e-42cb-ac59-7acdca56d798

Note: If all the Tacacs+ server configurations are deleted, the external auth will not get disabled on its own. The external auth needs to be disabled explicitly (as in 4.24.1.2).

4.24.2 Accounting / Logging to Tacacs+ Server

4.24.2.1 Update Logging Interval

The accounting logs are collected at periodic intervals and sent to Tacacs+. The default is set to 30 secs. The logging interval can be changed at any time using a rest call.

PUT	/cisco-vts/external-authentication/external-auth-enabled-info/logging-interval	Change the log collection interval

Sample Payload

{

"cisco-vts:logging-interval": 30

}

4.24.2.2 Enable Accounting

For accounting logs to be sent to the external AAA server, the accounting needs to be enabled.

Please note: We support only 1 AAA server for accounting at a time, so if there is already one enabled, the rest consumer needs to disable it. And then enable the tacacs server

PATCH	/cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/<tacacs-server-uuid>	Enable accounting for a tacacs server

Sample Payload

cisco-vts/external-authentication/external-auth-modes/external-auth-mode/tacacs-external-auth-mode/tacacs-server-credentials/tacacs-server-credential/0ca6434c-408e-42cb-ac59-7acdca56d79d


```
{
   "cisco-vts:tacacs-server-credential": {
      "accounting": {
        "accounting-enabled": [null]
      }
   }
}
```


5 Debugging and troubleshooting

5.1 VTS

5.1.1 Services

There are 3 VTS specific services on VTC :

1) ncs:. The controller and handles NB API etc.
2) tomcat7: Handles the user interface
3) vtsWebServer: Handles Auto Discovery API and Openstack Plugin installation etc.

These services are managed via the Linux service commands, e.g. “service <service-name> status” to query a service.

5.1.2 Logs

The VTC logs are collected in the /var/log/ncs directory

6 Appendix

This appendix explains the YANG model for the VTS REST API.

6.1 VTS Types

```
module cisco-vts-types {

 namespace "http://cisco.com/ns/yang/vts/types";

 prefix vts-types;

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, it defines types used in
   the context of VTS.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

 revision "2015-06-13" {
      description
    "Initial revision.";
 }

 /*
  * Typedefs
  */
 typedef uuid {
  type string {
    length "32|36";
  }
 }
 typedef string128 {
  type string {
    length "0..128" {
      error-message "This string cannot exceed 128 characters.";
    }
  }
 }
 typedef string255 {
  type string {
    length "0..255" {
      error-message "This string cannot exceed 255 characters.";
```

```
    }
  }
}
typedef string512 {
  type string {
    length "0..512" {
      error-message "This string cannot exceed 512 characters.";
    }
  }
}
typedef string15 {
  type string {
    length "0..15" {
      error-message "This string cannot exceed 15 characters.";
    }
  }
}
typedef vlan-id {
  type uint32 {
    range "2..4094";
  }
}
typedef subinterface-id {
  type uint32 {
    range "1..511";
  }
}
typedef vni {
  type uint32 {
    range "4096..65535";
  }
}
typedef ip-mask {
  type string {
    pattern '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
    + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
    + '/(([0-9])|([1-2][0-9])|(3[0-2]))';
  }
}
typedef vts-multicast-ip {
   type string {
     pattern '(239\.)' +
             '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){2}' +
             '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
  }
 }

typedef mac {
  type string {
    pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
```

```
    }
  }

  typedef unknown-mac {
    type string {
      pattern 'unknown-.{32,36}';
    }
  }

  typedef fex-identifier {
    type uint8 {
      range "100..199";
    }
  }
}
```

6.2 VTS Identities

```
module cisco-vts-identities {

  namespace "http://cisco.com/ns/yang/vts/identities";

  prefix vts-ids;

  import tailf-common {
    prefix tailf;
  }

  organization "Cisco Systems, Inc.";

  contact
    "Cisco Systems, Inc.
     Customer Service

     Postal: 170 West Tasman Drive
     San Jose, CA 95134

     Tel: +1 800 533-NETS";

  description
    "This module contains a collection of YANG definitions
     for Cisco's VTS's management. Specifically, it defines identities used in
     the context of VTS.

     Copyright (c) 2015 by Cisco Systems, Inc.
     All rights reserved.";
```

```
revision "2017-04-17" {
    description
      "VTS 2.6
       Enhancements:
         - Added identities virtual-switch type and vtf-links.
         - Added identities for l2 virtual switch group type.
     ";

}

revision "2016-08-30" {
  description
    "VTS 2.3.1
     Enhancements:
       - Added identities for custom and vpc device groups
     Corrections:
   ";

}

revision "2016-07-15" {
  description
    "VTS 2.3.0
     Enhancements:
       - Added identity for VMM version Openstack Liberty(RHEL)
       - Added identities for VTF installation mode for Kilo
       - Added identities for IPv6 address assignment
       - Added identities for VTF deployment mode
     Corrections:
   ";

}

revision "2015-06-13" {
      description
    "Initial revision.";
}

/*
 * Identities
 */
identity network-type {
  description "Based identity from which network types are " +
              "derived.";
}
identity vlan {
  base "network-type";
  description "Vlan network.";
  tailf:info "Vlan network.";
}
```

```
identity trunk {
  base "network-type";
  description "Trunk network.";
  tailf:info "Trunk network.";
}
identity gre {
  base "network-type";
  description "GRE network.";
  tailf:info "GRE network.";
}
identity vxlan {
  base "network-type";
  description "VXLAN network.";
  tailf:info "VXLAN network.";
}
identity flat {
    base "network-type";
    description "Flat network.";
    tailf:info "Flat network.";
}

identity entity-status {
  description "Based identity from which entities' status are " +
              "derived.";
}
identity active {
  base "entity-status";
  description "Entity is active.";
  tailf:info "Entity is active.";
}
identity down {
  base "entity-status";
  description "Entity is down.";
  tailf:info "Entity is down.";
}
identity build {
  base "entity-status";
  description "Entity is being built.";
  tailf:info "Entity is being built.";
}

identity binding-type {
  description "Based identity from which port binding types " +
              "are derived.";
}
identity ovs {
  base "binding-type";
  description "Ovs port binding type.";
  tailf:info "Ovs port binding type.";
}
```

```
identity unbound {
  base "binding-type";
  description "No binding provided.";
  tailf:info "No binding provided.";
}
identity vhostuser {
  base "binding-type";
  description "Vhostuser port binding type..";
  tailf:info "Vhostuser port binding type.";
}
identity hw-veb  {
  base "binding-type";
  description "hw-veb supporting sriov port binding type..";
  tailf:info "hw-veb supporting sriov port binding type.";
}
// TODO:   Consider semantics of a "failed" type. Mind we have port-status.
identity binding-failed {
  base "binding-type";
  description "Port binding failed.";
  tailf:info "Port binding failed.";
}
identity switch-platform {
  description "Based identity from which switch platform types " +
              "are derived.";
}
identity N3K {
  base "switch-platform";
  description "Cisco N3K.";
  tailf:info "Cisco N3K.";
}
identity N5K {
  base "switch-platform";
  description "Cisco N5K.";
  tailf:info "Cisco N5K.";
}
identity N6K {
  base "switch-platform";
  description "Cisco N6K.";
  tailf:info "Cisco N6K.";
}
identity N7K {
  base "switch-platform";
  description "Cisco N7K.";
  tailf:info "Cisco N7K.";
}
identity N9K {
  base "switch-platform";
  description "Cisco N9K.";
  tailf:info "Cisco N9K.";
}
```

```
identity ASR9K {
  base "switch-platform";
  description "Cisco ASR9K.";
  tailf:info "Cisco ASR9K.";
}
identity VTSR {
  base "switch-platform";
  description "Cisco VTSR.";
  tailf:info "Cisco VTSR.";
}
identity NOTAPPLICABLE {
  base "switch-platform";
  description "Platform unknown to VTS.";
  tailf:info "Platform unknown to VTS.";
}

identity switch-role {
  description "Based identity from which switch role types " +
              "are derived.";
}
identity leaf {
  base "switch-role";
  description "The device is acting as a leaf.";
  tailf:info "Leaf.";
}
identity spine {
  base "switch-role";
  description "The device is acting as a spine.";
  tailf:info "Spine.";
}
identity border-leaf {
  base "switch-role";
  description "The device is acting as a border-leaf.";
  tailf:info "Border-eaf.";
}
identity dci {
  base "switch-role";
  description "The device is acting as a DCI.";
  tailf:info "DCI.";
}
identity bgp-rr {
  base "switch-role";
  description "The device is acting as a BGP RR.";
  tailf:info "BGP-RR.";
}
identity spine-rr {
  base "switch-role";
  description "The device is acting as a spine and also as a RR";
}
identity unmanaged {
```

```
  base "switch-role";
  description "The device is not managed by VTS";
}

identity network-layer {
  description "Based identity from which network layers are " +
              "derived.";
}
identity l2 {
  base "network-layer";
  description "Layer 2.";
  tailf:info "Layer 2.";
}
identity l3 {
  base "network-layer";
  description "Layer 3.";
  tailf:info "Layer 3.";
}

identity l2-mode {
  description "Based identity from which layer 2 modes are " +
              "derived.";
}
identity l2-access {
  base "l2-mode";
  description "Access mode.";
  tailf:info "Access.";
}
identity l2-trunk {
  base "l2-mode";
  description "Trunk mode.";
  tailf:info "Trunk.";
}

identity server-type {
  description "Based identity from which server types are " +
              "derived.";
}
identity baremetal {
  base "server-type";
  description "Baremetal server.";
  tailf:info "Baremetal.";
}
identity virtual-server {
  base "server-type";
  description "Virtual server.";
  tailf:info "Virtual server.";
}
identity fex {
  base "server-type";
```

```
  description "Fabrix extender.";
  tailf:info "Fabric extender.";
}

//DO NOT USE THIS IN CODE - LEAVING FOR UPGRADE
identity vtf {
  base "server-type";
  description "VTF";
  tailf:info "VTF.";
}

//DO NOT USE THIS IN CODE - LEAVING FOR UPGRADE
identity vtf-physical-uplink {
  base "server-type";
  description "Uplink TOR of a VTF";
  tailf:info "Uplink TOR of a VTF";
}

identity switch-type{
  description "Base identity of virtual switch type in case of virtual server";
}

identity not-defined-st{
  base "switch-type";
  description "Virtual server switch type is not set yet.";
}
identity ovs-st{
  base "switch-type";
  description "Virtual server switch type is set as ovs.";
}

identity dvs-st{
  base "switch-type";
  description "Virtual server switch type is set as dvs.";
}

identity vtf-vtep-st{
  base "switch-type";
  description "Virtual server switch type is set as vtep.";
}

identity vtf-l2-st{
  base "switch-type";
  description "Virtual server switch type is set as vtf l2.";
}

identity vtf-link {
  description "Base identity of VTF link, identifying uplink to ToR or VTSr link.";
}
```

```
identity vtf-tor-link {
  base "vtf-link";
  description "Identifies VTF host connection to ToR. This will be applicable to
               V-side in multicast mode";
}

identity vtf-vtsr-link {
  base "vtf-link";
  description "Identifies VTF host connection to VTSR. This will be applicable in
  VTF as VTEP, VTF as L2";
}

identity l2-vs-group-type{
  description "Base identity of L2 virtual switch group type";
}

identity ovs-l2-vs-group{
  base "l2-vs-group-type";
  description "L2 virtual switch group - hosts managed by OVS";
  tailf:info "L2 virtual switch group - hosts managed by OVS";
}

identity vtc-interface-type {
  description "VTC interface type -- management (default) or underlay.";
}

identity management {
  base "vtc-interface-type";
  description "Management interface.";
  tailf:info "Management interface.";
}

identity underlay {
  base "vtc-interface-type";
  description "Underlay interface.";
  tailf:info "Underlay interface.";
}

identity dvs-l2-vs-group{
  base "l2-vs-group-type";
  description "L2 virtual switch group - hosts managed by DVS";
  tailf:info "L2 virtual switch group - hosts managed by DVS";
}

identity vtf-l2-vs-group{
  base "l2-vs-group-type";
  description "L2 virtual switch group - hosts managed by vtf-l2 switch";
  tailf:info "L2 virtual switch group - hosts managed by vtf-l2 switch";
}
```

```
  identity peer-type {
    description "Based identity from which connection peer types " +
                "are derived.";
  }
  identity server {
    base "peer-type";
    description "Server.";
    tailf:info "Server.";
  }
  identity fabric {
    base "peer-type";
    description "Fabric node.";
    tailf:info "Fabric node.";
  }


  identity port-channel-type {
    description "Based identity from which port channel types " +
                "are derived.";
  }
  identity vpc {
    base "port-channel-type";
    description "Virtual port channel type.";
    tailf:info "Virtual port channel.";
  }
  identity no-port-channel {
    base "port-channel-type";
    description "None.";
    tailf:info "None.";
  }

  identity control-plane-protocol {
    description "Base identity from which control plane protocols " +
                "for learning end points' addresses are derived.";
  }
  identity bgp-evpn {
    base "control-plane-protocol";
    description "BGP-EVPN control plane protocol.";
    tailf:info "BGP-EVPN.";
    reference
      "http://www.cisco.com/c/en/us/products/collateral/routers/asr-9000-series-
aggregation-services-routers/whitepaper_c11-731864.pdf";
  }
  identity flood-and-learn {
    base "control-plane-protocol";
    description "The flood-and-learn control plane for learning " +
                "the addresses of end points.";
    tailf:info "Flood-and-learn.";
  }
```

```
identity packet-replication-mode {
  description "Base identity from which modes of packet " +
              "replication are derived. Packet replication is " +
              "required for packets sent to multiple recipients";
}
identity multicast {
  base packet-replication-mode;
  description "Packet replication via multicast.";
  tailf:info "Multicast.";
}
identity ingress-replication {
  base packet-replication-mode;
  description "Packet replication by ingress-replication, whereby " +
              "a unicast packet is generated per recipient.";
  tailf:info "Ingress replication.";
}

identity route-reflector-mode {
  // TODO: improve description
  description "Base identity from which modes of route " +
              "reflectors.";
}
identity in-line-rr {
  // TODO: improve description
  base route-reflector-mode;
  description "In-line route reflectors.";
  tailf:info "In-line.";
}
identity global-rr {
  // TODO: improve description
  base route-reflector-mode;
  description "Global route reflectors.";
  tailf:info "Global.";
}

identity l3-distribution-mode {
  tailf:info "Base identity from which l3 distribution types are derived.";
}
identity centralized-l3 {
  base l3-distribution-mode;
  description "Centralized L3.";
  tailf:info "Centralized L3.";
}
identity decentralized-l3 {
  base l3-distribution-mode;
  description "Decentralized L3.";
  tailf:info "Decentralized L3.";
}

identity l2-distribution-mode {
```

```
  tailf:info "Base identity from which l2 distribution types are derived.";
}
identity decentralized-l2 {
  base l2-distribution-mode;
  description "Decentralized L2.";
  tailf:info "Decentralized L2.";
}

identity op-type {
  tailf:info "Base identity from which operation types are derived.";
}
identity create {
  base op-type;
  description "Create operation.";
  tailf:info "Create operation.";
}
identity delete {
  base op-type;
  description "Delete operation.";
  tailf:info "Delete operation.";
}

identity transaction-type {
  tailf:info "Base identity from which transaction types are derived.";
}
identity port {
  base transaction-type;
  description "Port operation.";
  tailf:info "Port operation.";
}
identity interface {
  base transaction-type;
  description "Interface operation.";
  tailf:info "Interface operation.";
}

identity device-group-type {
  description "Based identity from which the device group types are derived.";
}
identity dvs {
  base "device-group-type";
  description "DVS.";
  tailf:info "DVS.";
}
identity vpc-group {
  base "device-group-type";
  description "VPC.";
  tailf:info "VPC.";
}
identity esi-group {
```

```
    base "device-group-type";
    description "ESI.";
    tailf:info "ESI.";
  }
  identity device-interface-group {
    base "device-group-type";
    description "Device Interface Group.";
    tailf:info "Device Interface Group.";
  }
  identity fex-interface-group {
    base "device-group-type";
    description "Device Interface Group.";
    tailf:info "Device Interface Group.";
  }
  identity dvs-device-interface-group {
    base "device-group-type";
    description "DVS Device Interface Group.";
    tailf:info "DVS Device Interface Group.";
  }
  identity dvs-fex-interface-group {
    base "device-group-type";
    description "DVS Fex Interface Group.";
    tailf:info "DVS Fex Interface Group.";
  }
  identity custom {
    base "device-group-type";
    description "Custom.";
    tailf:info "Custom.";
  }

  identity device-group-device-type {
    description "Based identity from which the device in the device group types are
derived.";
  }
  identity v-vtep {
    base "device-group-device-type";
    description "Virtual device type for VTF.";
    tailf:info "Virtual device type for VTF.";
  }
  identity p-vtep {
    base "device-group-device-type";
    description "Physical device type for TOR.";
    tailf:info "Physical device type for TOR.";
  }
  identity service-policy-mode{
    tailf:info "Base identity from which service policy modes are derived.";
  }
  identity no-policy-mode{
    base service-policy-mode;
    description "No service policy mode.";
```

```
  tailf:info "No service policy mode.";
}
identity network-to-epg-mode{
  base service-policy-mode;
  description "Network mapped to Endpoint Group.";
  tailf:info "Network mapped to Endpoint Group.";
}

identity setup-state{
  description "Base identity for the states in which a component can be during setup";
  tailf:info "Base identity for the states in which a component can be during setup";
}
identity setup-state-completed{
  base setup-state;
  description "A state to indicate that component has been setup";
  tailf:info "A state to indicate that component has been setup";
}
identity setup-state-initial{
  base setup-state;
  description "A state to indicate that component has not been setup";
  tailf:info "A state to indicate that component has not been setup";
}
identity setup-state-skipped{
  base setup-state;
  description "A state to indicate that component setup has been skipped";
  tailf:info "A state to indicate that component setup has been skipped";
}

identity vmm-type {
  tailf:info "Base identity for vmm type.";
}
identity openstack {
  base "vmm-type";
  tailf:info "An openstack vmm.";
}
identity vcenter {
  base "vmm-type";
  tailf:info "A vcenter vmm.";
}

identity vmm-version {
  tailf:info "Base identity for vmm version.";
}
identity openstack-version {
  base "vmm-version";
  tailf:info "Base identity for openstack version.";
}
identity openstack-icehouse {
  base "openstack-version";
  tailf:info "Openstack icehouse version.";
```

```
}
identity openstack-juno {
  base "openstack-version";
  tailf:info "Openstack juno version.";
}
identity openstack-kilo {
  base "openstack-version";
  tailf:info "Openstack kilo version.";
}
identity openstack-liberty-centos {
  base "openstack-version";
  tailf:info "Openstack Liberty(CentOS) version.";
}
identity openstack-liberty-rhel {
    base "openstack-version";
    tailf:info "Openstack Liberty(RHEL) version.";
}
identity openstack-newton {
    base "openstack-version";
    tailf:info "Openstack Newton version.";
}
identity vcenter-version {
  base "vmm-version";
  tailf:info "Base identity for vcenter version.";
}
identity vcenter-6 {
  base "vcenter-version";
  tailf:info "Vcenter 6.";
}
identity vcenter-5.5 {
   base "vcenter-version";
   tailf:info "Vcenter 5.5.";
}

identity vmm-registration-state{
  description "Base identity for the vmm registration.";
  tailf:info "Base identity for the vmm registration.";
}
identity vmm-registration-completed{
  base vmm-registration-state;
  description "A state to indicate that vmm registration is complete.";
  tailf:info "A state to indicate that vmm registration is complete.";
}
identity vmm-registration-initial{
  base vmm-registration-state;
  description "A state to indicate that vmm registration has not been started.";
  tailf:info "A state to indicate that vmm registration has not been started.";
}
identity vmm-registration-failed{
  base vmm-registration-state;
```

```
  description "A state to indicate that vmm registration has failed.";
  tailf:info "A state to indicate that vmm registration has failed.";
 }
 identity vmm-registration-in-progress{
  base vmm-registration-state;
  description "A state to indicate that vmm registration is in progress.";
  tailf:info "A state to indicate that vmm registration is in progress.";
 }

 identity server-capability {
   description "Base identity for the capability of the server.";
   tailf:info "Base identity for the capability of the server.";
  }
  identity no-virtual-switch {
   base server-capability;
   description "Server has no virtual-switch capability. " +
               "Server can only be connected to physical switches.";
   tailf:info "Server has no virtual-switch capability.";
  }
  identity virtual-switch {
   base server-capability;
   description "Server has virtual-switch capability.";
   tailf:info "Server has virtual-switch capability.";
  }
  identity install-status {
   description "Status of the installation.";
   tailf:info "Status of the installation.";
  }
  identity not-applicable {
   base install-status;
   description "Nothing to be installed.";
   tailf:info "Nothing to be installed.";
  }
  identity vtf-installed {
   base install-status;
   description "VTF installed.";
   tailf:info "VTF installed.";
  }
  identity host-agent-installed {
   base install-status;
   description "Host Agent installed.";
   tailf:info "Host Agent installed.";
  }
  identity vtf-host-agent-installed {
   base install-status;
   description "VTF and host agent installed.";
   tailf:info "VTF and host agent installed.";
  }
  identity vtf-install-failed {
   base install-status;
```

```
  description "VTF installation failed.";
  tailf:info "VTF installation failed.";
}
identity host-agent-install-failed {
  base install-status;
  description "Host Agent installation failed.";
  tailf:info "Host Agent installation failed.";
}
identity vtf-host-agent-install-failed {
  base install-status;
  description "VTF and host agent installation failed.";
  tailf:info "VTF and host agent installation failed.";
}
identity vtf-installation-in-progress {
  base install-status;
  description "VTF installation in progress.";
  tailf:info "VTF installation in progress.";
}
identity host-agent-installation-in-progress {
  base install-status;
  description "Host Agent installation in progress.";
  tailf:info "Host Agent installation in progress.";
}
identity vtf-host-agent-installation-in-progress {
  base install-status;
  description "VTF and host agent installation in progress.";
  tailf:info "VTF and host agent installation in progress.";
}

// Identities for VTF installation mode for Kilo
identity vtf-mode {
  description "VTF Installation mode for Kilo";
  tailf:info "VTF Installation mode for Kilo";
}
identity VM {
  base vtf-mode;
  description "Install VTF as a VM.";
  tailf:info "Install VTF as a VM.";
}
identity vhost {
  base vtf-mode;
  description "Install VTF as a host process on compute.";
  tailf:info "Install VTF as a host process on compute.";
}

// Identities for IPv6 address assignment
identity ipv6-address-assignment {
 description "Based identity from which different mechanisms " +
    "for assignement of IPv6 addresses.";
}
```

```
identity slaac {
  base "ipv6-address-assignment";
  description "Stateless address autoconfiguration.";
  tailf:info "Stateless address autoconfiguration.";
}
identity dhcpv6-stateful {
  base "ipv6-address-assignment";
  description "DHCP Stateful.";
  tailf:info "DHCP Stateful.";
}
identity dhcpv6-stateless {
  base "ipv6-address-assignment";
  description "DHCP Stateless.";
  tailf:info "DHCP Stateless.";
}

// Identities for VTF deployment mode
identity vtf-deployment-mode {
  description "Base identity for the VTF installation mode.";
  tailf:info "Base identity for the VTF installation mode.";
}
identity vtf-vhost {
  base vtf-deployment-mode;
  description "VTF running in vhost mode.";
  tailf:info "VTF running in vhost mode.";
}
identity vtf-as-vm {
  base vtf-deployment-mode;
  description "VTF running as a virtual machine.";
  tailf:info "VTF running as a virtual machine.";
}

// Identities for NIC bonding modes
identity nic-bond-mode {
  description "Base identity for the nic bonding modes.";
  tailf:info "Base identity for the nic bonding modes.";
}
identity active-backup {
  base nic-bond-mode;
  description "Active Backup nic bonding mode.";
  tailf:info "Active Backup nic bonding mode.";
}
identity balance-xor-l23 {
  base nic-bond-mode;
  description "Balance xor l23 nic bonding mode.";
  tailf:info "Balance xor l23 nic bonding mode.";
}
identity balance-xor-l34 {
  base nic-bond-mode;
  description "Balance xor l34 nic bonding mode.";
```

```
    tailf:info "Balance xor l34 nic bonding mode.";
  }
  identity balance-xor-l2 {
    base nic-bond-mode;
    description "Balance xor l2 nic bonding mode.";
    tailf:info "Balance xor l2 nic bonding mode.";
  }

  // Identities for pci driver
  identity pci-driver-type {
    description "Base identity for pci driver type.";
    tailf:info "Base identity for pci driver type.";
  }
  identity uio_pci_generic {
    base pci-driver-type;
    description "uio_pci_generic pci driver type.";
    tailf:info "uio_pci_generic pci driver type.";
  }
  identity vfio-pci {
    base pci-driver-type;
    description "vfio-pci driver type.";
    tailf:info "vfio-pci driver type.";
  }
}
```

6.3 VTS

```
module cisco-vts {

  namespace "http://cisco.com/ns/yang/vts";

  prefix vts;


  import tailf-common {
    prefix tailf;
  }

  include cisco-vts-common {
    revision-date 2017-06-26;
  }
  include cisco-vts-system-config {
    revision-date 2017-05-30;
  }
  import cisco-vts-types {
    prefix vts-types;
  }
  include cisco-vts-network {
    revision-date 2015-06-12;
```

```
}
include cisco-vts-inventory {
  revision-date 2016-08-30;
}
include cisco-vts-custom-template {
  revision-date 2015-06-15;
}
include cisco-vts-infrastructure-policy {
  revision-date 2017-04-28;
}
include cisco-vts-device-operational-data {
  revision-date 2015-06-15;
}

include cisco-vts-uuid-server {
  revision-date 2017-04-20;
}

include cisco-vts-vpns {
  revision-date 2016-12-20;
}

include cisco-vts-authentication {
  revision-date 2016-09-15;
}

include cisco-vts-mvmm-policy {
  revision-date 2017-01-26;
}

include cisco-vts-security-groups {
  revision-date 2017-10-31;
}

include static-route {
  revision-date 2017-06-22;
}

import tailf-ncs {
  prefix ncs;
}

import ietf-inet-types {
  prefix inet;
}

import ietf-yang-types {
    prefix ietf-yang;
  }
```

```
import cisco-vts-identities {
  prefix vts-ids;
}

import resource-allocator {
  prefix ralloc;
}
import vni-allocator {
  prefix vnialloc;
}

import cisco-vts-templates {
  prefix templates;
}

import cisco-vts-templates-target-mapping{
  prefix cisco-vts-templates-target-mapping;
}

include cisco-vts-service-policy {
  revision-date 2017-07-21;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

  revision "2017-09-19" {
      description
        "VTS 2.6
         Enhancements:
             - Added leaf install-vmm-components to VMMs model
       ";
   }

 revision "2017-08-26" {
```

```
    description
      "VTS 2.6
       Enhancements:
          - Support of ARP suppression at network level
     ";

 }

revision "2017-06-26" {
  description
    "VTS 2.6
     Enhancements:
        - Added trunk model and trunk details for ports.
   ";

}
revision "2017-03-13" {
  description
    "VTS 2.5
     Enhancements:
        - Removed the constraint for only allow OpenStack to create Shared Network.
   ";

}

revision "2017-03-10" {
  description
    "VTS 2.5
     Enhancements:
        - Added multi vmm policies.
        - Added VTS-SHARED-SUBNETS
   ";

}

revision "2016-12-15" {
  description
       "VTS 2.4.1
        Enhancements:
          - Added leaf overlay-router-vpn-id to routers container that leaf-ref
            the Global Route Leaking Service definition in submodule cisco-vts-vpns
        Corrections:
          - None
       ";
}


revision "2016-11-15" {
  description
    "VTS 2.4
```

```
      Enhancements:
        - Added leaf router-gateway-ipv6-address under list router.
      Corrections:
        - Under router interface list, leaves ip-address and
            router-gateway-ip-address modified to support both v4
            and v6. Previously, only v4 was supported.
        - Added constraint so that a network can only support one
            subnet per ip version.
        - Added constraint so that the subnet cidr and gateway match
            the provided ip-version.
    ";
  }


  revision "2016-08-30" {
    description
      "VTS 2.3.1
      Enhancements:
          - Added leaf vpn-id to networks container
          - Added group VTS-VPNS to cisc-vts container
      Corrections:
          - Removed unique constraint for subnetworks
          - Relaxed must constraint in the leaf interface-name
    ";

  }

  revision "2016-07-15" {
    description
      "VTS 2.3.0
      Enhancements:
        - Added leaf vrf-name under list router. The name must be unique
            across tenants. A must statement enforces this.
        - Added identities for template targets
        - Added leaf vtf-deployment under list vmm
        - Added groupings TEMPLATES-DEVICE-GROUP and  TEMPLATE-
TARGET-MAPS
            used under cisco-vts
      Corrections:
        - Removed must constraint for subnet name as it can be any string
        - Added unique statement for leaf name under list router
        - Added must statement so that the name of a subnetwork is unique
            within the scope of a network
        - Added must statement so that an external network cannot contain
            subnetworks attached to a router
        - Modified types used for ip addresses from generic to v4 to reflect
            accurately the expected type
    ";

  }
```

```
revision "2015-06-15" {
  description
    "Enhancements:
     Added cisco-vts container which serves as
     root for all cisco-vts objects.
     Added tenants and toplogies.
     Included cisco-vts-system-config submodule.
     Corrections:
    ";
}

revision "2015-02-28" {
 description
    "Initial revision.";
}

identity tagging-requirement-type {
  description "Based identity from which tagging requirements " +
              "types are derived.";
}
identity optional {
  base "tagging-requirement-type";
  description "Tagging is optional.";
  tailf:info "Tagging is optional.";
}
identity mandatory {
  base "tagging-requirement-type";
  description "Tagging is mandatory.";
  tailf:info "Tagging is mandatory.";
}

/*
 * Identities
 */
identity network-tag-type {
  description "Based identity from which network tag types are " +
              "derived.";
}
identity vlan-tag {
  base "network-tag-type";
  description "Vlan tagging.";
  tailf:info "Vlan tagging.";
}

identity tag-scope-type {
  description "Based identity from which tag scope types are " +
               "derived.";
}
identity global {
```

```
  base "tag-scope-type";
  description "The scope of the tag is global.";
  tailf:info "The scope of the tag is global.";
}
identity port {
  base "tag-scope-type";
  description "The scope of the tag is the port.";
  tailf:info "The scope of the tag is the port.";
}

identity port-creation {
  tailf:info "Base identity from which port creation options are derived.";
}
identity none {
  base "port-creation";
  tailf:info "No port creation.";
}
identity create {
  base "port-creation";
  tailf:info "Create port.";
}

identity router-gateway-creation {
  tailf:info "Base identity from which router gateway creation options are derived.";
}
identity unset {
  base "router-gateway-creation";
  tailf:info "No router gateway creation.";
}
identity set {
  base "router-gateway-creation";
  tailf:info "Set router gateway.";
}

// VMM Modes
identity vmm-mode {
  description "Base identity for mode of VMM";
}

identity trusted {
  base "vmm-mode";
  description "Trusted mode.";
  tailf:info "Trusted mode.";
}

identity untrusted {
  base "vmm-mode";
  description "Untrusted mode.";
  tailf:info "Untrusted mode.";
}
```

```
// VMM REST protocol
identity rest-protocol {
  description "Base identity for protocol used for calling REST APIs of VMM";
}

identity http {
  base "rest-protocol";
  description "http protocol.";
  tailf:info "http protocol.";
}

identity https {
  base "rest-protocol";
  description "https protocol.";
  tailf:info "https protocol.";
}

grouping VXLAN-NETWORK-ID {
  choice vxlan-network-identifier {
    case user-assigned {
      leaf vni {
        description "Vni to use for this virtual entity";
        type vts-types:vni;
        //must "1 = count(/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range[" +
        //     "vnialloc:start <= current()/../vni " +
        //     "and vnialloc:end >= current()/../vni])" {
        //  error-message "The vni must belong to a vni range.";
        //  tailf:dependency "/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range";
        //}
      }
    } // case user-assigned
    case vts-allocated {
      leaf vts-allocated-vni {
        description "Vni allocated by VTC for this virtual entity. The
                     allocation can be dynamic or statically pre-allocated
                     by the user.";
        type vts-types:vni;
        //must "1 = count(/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range[" +
        //     "vnialloc:start <= current()/../vts-allocated-vni " +
        //     "and vnialloc:end >= current()/../vts-allocated-vni])" {
        //  error-message "The vni must belong to a vni range.";
        //  tailf:dependency "/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range";
        //}
      }
    } // case vts_allocated
  } // choice vxlan_network_identifier
} // grouping VXLAN-NETWORK-ID

grouping MULTICAST-ADDRESS-IDENTIFIER {
```

```
 choice multicast-ip-address {
   case user-assigned {
    leaf multicast-address {
      description "Multicast address to use for this virtual entity";
      type vts-types:vts-multicast-ip;
      }
   }// case user-assigned
   case vts-allocated {
    leaf vts-allocated-multicast{
      description "Multicast allocated by VTC for this virtual entity. The
                   allocation can be dynamic or statically pre-allocated
                   by the user.";
      type inet:ipv4-address;
    }
   }//case vts-allocated-multicast
  }//choice multicast-ip-address
 }//grouping  MULTICAST-ADDRESS-IDENTIFIER


grouping VTS-PORTS {
  container ports  {
    description "Set of ports.";
    tailf:info "Ports.";
    list port {
      description "List of ports.";
      tailf:info "Port.";

      ncs:servicepoint vts-port-servicepoint;

      key id;
      leaf id {
        description "Port identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }

      uses ncs:service-data;
      uses vts-types:MODIFIED-TIME-ELEMENT;

      leaf network-id {
        description "Identifier of the associated network.";
        tailf:info "ID of the associated network.";
        type vts-types:uuid;
        mandatory "true";
      }

      uses vts:COMMON-PORT;

      leaf binding-vif-type {
        description "Binding VIF type.";
```

```
  tailf:info "Binding VIF type.";
  type identityref {
    base "vts-ids:binding-type";
  }
  default vts-ids:ovs;
}

container vhostuser-details {
  leaf vhostuser-mode {
    description "vhost user mode.";
    tailf:info "vhostuser mode";
    type vts-types:string128;
  }

  leaf vhostuser-socket {
    tailf:info "vhostuser socket path";
    type vts-types:string128;
  }
}

leaf status {
 description "Port status.";
  tailf:info "Status.";
  type identityref {
    base "vts-ids:entity-status";
  }
  mandatory true;
}

choice vlan-network-identifier {
  case user-assigned {
    leaf vlan-id {
      description "Vlan id to use for this port.";
      tailf:info "Vlan for the port.";
      type vts-types:vlan-id;
    }
  } // case user-assigned
  case vts-allocated {
    leaf vts-allocated-vlan-id {
      description "Vlan id allocated by VTC for this port. The allocation can be
                   dynamic or statically pre-allocated by the user.";
      tailf:info "Vlan id allocated for the port.";
      type vts-types:vlan-id;
    }
  } // case vts-allocated
} // choice vlan_network_identifier

choice device-interface-vlan-network-identifier {
  case user-assigned {
    leaf interface-vlan-id {
```

```
            description "Device Interface Vlan id to use for this port.";
            tailf:info "Device Interface Vlan for the port.";
            type vts-types:vlan-id;
           }
          } // case user-assigned
          case vts-allocated {
           leaf vts-allocated-interface-vlan-id {
            description "Device Interface Vlan id allocated by VTC for this port. The
allocation can be
                        dynamic or statically pre-allocated by the user.";
            tailf:info "Device Interface Vlan id allocated for the port.";
            type vts-types:vlan-id;
           }
          } // case vts-allocated
        } // choice vlan_network_identifier

        //Baremetal-specific fields
        leaf type {
          description "Type of server port.";
          tailf:info "Type of server port.";
          type identityref {
            base "vts-ids:server-type";
          }
          default vts-ids:virtual-server;
        }
        leaf ignore-trunk-vlan {
          description "Flag to ignore L2 config push to Tor ports";
          tailf:info "Flag to ignore L2 config push to Tor ports";
          type boolean;
          default false;
          //when "../type = 'vts-ids:baremetal'";
        }
        leaf tagging {
          description "Indicates whether a baremetal server tags " +
                      "the packets it sends.";
          tailf:info "Indicates whether a baremetal server tags " +
                     "the packets it sends.";
          type identityref {
            base "tagging-requirement-type";
          }
          //when "../type = 'vts-ids:baremetal'";
        }
        leaf network-tag {
          description "Type of network tag used by a baremetal " +
                      "server.";
          tailf:info "Type of network tag used by a baremetal " +
                      "server.";
          type identityref {
            base "network-tag-type";
          }
```

```
  default vlan-tag;
  //when "../type = 'vts-ids:baremetal'";
 }
 leaf tag-scope {
  description "Scope of the network tag used by a baremetal " +
              "server.";
  tailf:info "Scope of the network tag used by a baremetal " +
             "server.";
  type identityref {
   base "tag-scope-type";
  }
  default global;
  //when "../type = 'vts-ids:baremetal'";
 }
 container fixed-ips {
  description "Set of fixed IP addresses.";
  tailf:info "IP addresses.";

  list fixed-ip {
   // TODO: this list needs to be revisited
   // 1) is the IP address mandatory?
   // 2) subnet-id comes from OS as a list of subnets
   //    our template concatenates them into the subnet-id leaf.
   //    How is this used? Needs to be revisited
   description "List of fixed IP addresses.";
   tailf:info "Fixed IP address.";
   key ip-address;

   leaf ip-address {
    description "IP address.";
    tailf:info "IP address.";
    type inet:ip-address;
   }
   leaf subnet-id {
    description "Subnetwork identifier.";
    tailf:info "Subnetwork ID.";
    type vts-types:string128;
   }
  } // list fixed-ips
 } // container fixed-ips
 container binding-capabilities {
  description "List of port binding capabilities.";
  tailf:info "Binding capability.";
  leaf port-filter {
   description "Port filter.";
   tailf:info "Port filter.";
   type boolean;
   default "false";
  }
 }
```

```
    list connid {
      key id;
      ordered-by user;
      leaf id {
        tailf:info "Unique ID for a ToR, ToR port, server name and server port.";
        type leafref {
          path "/cisco-vts/uuid-servers/uuid-server/connid";
        }
        //type vts-types:uuid;
      }
    } // list connid
    leaf port-type {
      description "Regular, parent, subport";
      tailf:info "Regular, parent, subport";
      type identityref {
        base "vts-ids:port-type";
      }
      default "vts-ids:regular-port";
    }
    leaf parent-port {
      description "Parent port of subport";
      tailf:info "Parent port of subport";
      type leafref {
        path "../../port/id";
      }
    }

    list endpoint-groups {

      description "List of endpoint groups.";
      tailf:info "Endpoint Group.";

      key epg-id-ref;

      leaf epg-id-ref {
        description "EPG identifier.";
        tailf:info "EPG identifier.";
        type leafref {
          path "/cisco-vts/tenants/tenant/endpoint-groups/endpoint-group/id";
        }
      }

    } // list endpoint-groups

    leaf port-edit-timestamp {
      type ietf-yang:date-and-time;
      description "The timestamp when this port was edited last.";
      tailf:info  "The timestamp when this port was edited last.";
    }
  }  // list port
```

```
  } // container ports
} // grouping VTS_PORTS

grouping VTS-SUBNETWORKS {
  container subnetworks  {
    description "Set of subnetworks.";
    tailf:info "Subnetworks.";

    list subnetwork {
      description "List of subnetworks.";
      tailf:info "Subnetwork.";

      ncs:servicepoint vts-subnet-servicepoint;

      key id;

      leaf id {
        description "Subnetwork identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }
      uses ncs:service-data;

      must "(./ip-version = 6 and contains(./cidr, ':') and contains(./gateway-ip, ':')) or "
+
          "(./ip-version !=6 and contains(./cidr, '.') and contains(./gateway-ip, '.')) " {
        error-message "The cidr and gateway-ip format must match the ip-version.";
        tailf:dependency ".";
      }

      leaf network-id {
        description "Identifier of the associated network.";
        tailf:info "ID of the associated network.";
        type vts-types:uuid;
        mandatory "true";
      }

      uses COMMON-SUBNETWORK;

      container allocation-pools {
        description "Set of allocation pools of IP addresses.";
        tailf:info "Allocation pools.";
        list allocation-pool {
          description "List of allocation pools of IP addresses.";
          tailf:info "Allocation pool.";
          key start;
          // TODO: add "must" constraint so that pools do not overlap
          leaf start {
            description "Allocation pool's first IP address.";
            tailf:info "Pool's first IP address.";
```

```
          type inet:ip-address;
        }
        leaf end {
          description "Allocation pool's last IP address.";
          tailf:info "Pool's last IP address.";
          type inet:ip-address;
          must "../start <= ../end" {
            error-message "The end of the allocation pool " +
                          "must cannot be less than or " +
                          "equal to the start";
            tailf:dependency "../start";
          }
        }
      }  // allocation-pool
    }  // allocation-pools
    container dns-nameservers {
      description "Set of DNS name servers.";
      tailf:info "DNS name servers.";
      leaf-list dns-nameserver {
        description "List of DNS name servers.";
        tailf:info "DNS name server.";
        type inet:ip-address;
      }  // dns-nameserver
    }  // dns-nameservers
    container host-routes {
      description "Set of host routes.";
      tailf:info "Host routes";
      list host-route {
        description "List of host routes.";
        tailf:info "Host route.";
        key destination;
        leaf destination {
          description "Route's destination.";
          tailf:info "Destination.";
          type tailf:ipv4-address-and-prefix-length;
        }
        leaf nexthop {
          description "Route's nexthop gateway.";
          tailf:info "Nexthop gateway.";
          type inet:ip-address;
        }
      }  // host-route
    }  // host-routes
   } // subnetwork
  } // subnetworks
} // VTS_SUBNETWORKS

grouping VTS-SHARED-NETWORKS {
  container shared-networks {
    description "Set of shared networks in VTC";
```

```
    tailf:info "Shared Networks";

    list shared-network {
      key id;
      leaf id {
        description "Identifier of the shared network.";
        tailf:info "ID of the shared network.";
        type leafref {
          path "../../../tenants/tenant/topologies/topology/networks/network/id";
        }
      }
      leaf tenant-id {
        description "tenant id of the network";
        tailf:info "tenant id of the network";
        type leafref {
          path "../../../tenants/tenant/name";
        }
      }
      leaf topology-id {
        description "topology id of the network";
        tailf:info "topology id of the network";
        type leafref {
          path "../../../tenants/tenant/topologies/topology/id";
        }
      }
    }
  }
}

grouping VTS-SHARED-SUBNETS {
  container shared-subnets {
    description "Set of shared subnets in VTC";
    tailf:info "Shared Subnets";

    list shared-subnet {
      key id;
      leaf id {
        description "Identifier of the shared subnet.";
        tailf:info "ID of the shared subnet.";
        type leafref {
          path "../../../tenants/tenant/topologies/topology/subnetworks/subnetwork/id";
        }
      }
      leaf tenant-id {
        description "tenant id of the subnet";
        tailf:info "tenant id of the subnet";
        type leafref {
          path "../../../tenants/tenant/name";
        }
      }
```

```
      leaf topology-id {
        description "topology id of the subnet";
        tailf:info "topology id of the subnet";
        type leafref {
          path "../../../tenants/tenant/topologies/topology/id";
        }
      }
    } // shared-subnet
  }
} // VTS-SHARED-SUBNETS

grouping VTS-ROUTER-INTERFACE-ABSTRACTION {
  container l3-abstraction {
    description "VTS abstraction for L3 services";
    tailf:info "VTS L3 Abstraction.";
    container devices {
      description "L3 devices.";
      tailf:info "L3 devies.";
      list device {
        description "L3 devices.";
        tailf:info "L3 devies.";
        key name;
        leaf name {
          description "Device Name.";
          tailf:info "Device Name.";
          type vts-types:string128;
        }
        container l3-vnis {
          description "L3 Vnis.";
          tailf:info "L3 Vnis.";
          list l3-vni {
            description "L3 Vnis.";
            tailf:info "L3 Vnis.";
            ncs:servicepoint vts-router-interface-abstraction-servicepoint;
            uses ncs:service-data;
            key vni;
            leaf vni {
              description "L3 Vni.";
              tailf:info "L3 Vni.";
              type vts-types:vni;
            }
          }
        }
      }
    }
  }
}

grouping ENCAP-PROFILE-ABSTRACTION {
  container encap-profile-abstraction {
```

```
description "Encap profile Abstraction container.";
tailf:info "Encap profile Abstraction.";
container devices {
  description "VTS device.";
  tailf:info "Device.";
  list device {
    description "Device.";
    tailf:info "Device.";
    key name;
    leaf name {
      description "Device name.";
      tailf:info "Device name.";
      type vts-types:string128;
    }
    container interfaces {
      description "Device Interfaces.";
      tailf:info "Device Interface";
      list interface {
        description "Device Interfaces.";
        tailf:info "Device Interface";
        key name;
        leaf name {
          description "Interface name.";
          tailf:info "Interface name.";
          type vts-types:string128;
        }
        leaf encap-profile-name {
          description "Profile name.";
          tailf:info "Profile name.";
          type vts-types:string128;
        }
        leaf pool-name {
          description "Device Interface Pool Name.";
          tailf:info "Device Interface Pool Name.";
          type vts-types:string128;
        }
        leaf service-instance-id {
          description "Service Instance Id";
          tailf:info "Service Instance Id.";
          type vts-types:instance-id;
          default 0;
        }
        leaf in-use {
          description "Leaf to describe if this interface is in use.";
          tailf:info "Leaf to describe if this interface is in use.";
          type boolean;
          default false;
        }
      }
    }
```

```
      }
    }
  }
}

grouping VTS-SERVICE-ABSTRACTION {
  container abstractions {
    description "Top level VTS service abstraction.";
    tailf:info "Service Abstraction.";
    uses ENCAP-PROFILE-ABSTRACTION;
    uses VTS-ROUTER-INTERFACE-ABSTRACTION;
  }
}

grouping VTS-METADATA {
  container metadata {
    container router-vrf-name {
      description "List of routers with vrf-names";
      tailf:info "router names with associated vrf-names";
      list routers {
        description "List of Routers.";
        tailf:info "Routers.";
        key id;
        unique vrf-name;
        leaf id {
          description "Router id.";
          tailf:info "Router.";
          type vts-types:uuid;
        }
        leaf vrf-name {
          description "Vrf Name.";
          tailf:info "Vrf Name.";
          type vts-types:alphanumeric24;
        }
      }
    }
    container device-interface-pool {
      description "List of device interface pools with devices and interfaces";
      tailf:info "List of device interface pools";
      container devices {
        description "VTS device.";
        tailf:info "Device.";
        list device {
          description "Device.";
          tailf:info "Device.";
          key name;
          leaf name {
            description "Device name.";
            tailf:info "Device name.";
            type vts-types:string128;
```

```
      }
      container fexs {
       description "Device Fexs.";
       tailf:info "Device Fexs";
       list fex {
        description "Device Fex.";
        tailf:info "Device Fex";
        key id;
        leaf id {
         description "Fex Id.";
         tailf:info "Fex Id.";
         type vts-types:string128;
        }
        container peers {
         description "Peer Fexs.";
         tailf:info "Peer Fexs";
         list device {
          description "Peer Device.";
          tailf:info "Peer Device.";
          key name;
          leaf name {
           description "Device name.";
           tailf:info "Device name.";
           type vts-types:string128;
          }
          container fexs {
           description "Device Fexs.";
           tailf:info "Device Fexs";
           list fex {
            description "Device Fex.";
            tailf:info "Device Fex";
            key id;
            leaf id {
             description "Fex Id.";
             tailf:info "Fex Id.";
             type vts-types:string128;
            }
           }
          }
         }
        }
       }
      }
     }
    }
  }
  container multi-homing-groups {
   description "Multi Homing Group Metadata.";
   tailf:info "Multi Homing Group.";
   list multi-homing-group {
```

```
    description "Multi Homing Group.";
    tailf:info "Multi Homing Group.";
    key "group-name";
    leaf group-name {
      description "Multi Homing Group Name.";
      tailf:info "Multi Homing Group Name.";
      type vts-types:string128;
    }
    list hosts {
      key host-name;
      leaf host-name {
        description "Hostname of the server.";
        tailf:info "Hostname.";
        type vts-types:string128;
      }
    }
    container links {
      description "Links in the group.";
      tailf:info "Links in the group.";
      list link {
        description "Link in the MH group.";
        tailf:info "Link in the MH group.";
        key "device interface";
        leaf device {
          description "Device.";
          tailf:info "Device.";
          type vts-types:string128;
        }
        leaf interface {
          description "Device Interface.";
          tailf:info "Device Interface.";
          type vts-types:string128;
        }
      }
    }
  }
}
container in-use-links {
  description "Links in the group.";
  tailf:info "Links in the group.";
  list in-use-link {
    description "Link in the MH group.";
    tailf:info "Link in the MH group.";
    key "device interface";
    leaf device {
      description "Device.";
      tailf:info "Device.";
      type vts-types:string128;
    }
    leaf interface {
```

```
          description "Device Interface.";
          tailf:info "Device Interface.";
          type vts-types:string128;
        }
      }
    }
  }
}

grouping VTS-NETWORKS {
  container networks  {
    description "Set of networks.";
    tailf:info "Networks.";

    list network {
      description "List of Networks.";
      tailf:info "Network.";
      ncs:servicepoint vts-network-servicepoint;

      key id;

      leaf id {
        description "Network identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }

      uses ncs:service-data;

      leaf bridge-domain-id {
        description "Identifier of the associated network.";
        tailf:info "ID of the associated network.";
        type leafref {
          path "../../../bridge-domains/bridge-domain/id";
        }
        // TODO: Commenting out the mandatory flag for now. This needs
        // to be added back in once the network model is validated
        // and the use case for mandating is confirmed
        // mandatory "true";
        //
      }

      uses COMMON-NETWORK;

      leaf extend-to-core {
        description "Should network be extended to core (yes/no) or inherit the state
(none)";
        tailf:info "Should network be extended to core (yes/no) or inherit the state
(none)";
        type vts-types:yes-no-none-enum;
```

```
        default none;

        must "((current() = 'yes') or (current() = 'none') or ((current() = 'no') and
(current()/../router-external = 'false'))
              or ((current() = 'no') and (current()/../router-external = 'true' and (0 =
count(../../../ports/port[network-id=current()/../id])))))
                 "{
              error-message "Network cannot be L3 extended if ports are attached";
              tailf:dependency ".";
             }
      }

      leaf vpn-id {
        description "VPN object to use for stitching this network";
        type leafref {
          path "/cisco-vts/vpns/vpn/id";
        }
        must "1 = count(../../network[vpn-id=current()])" {
         error-message "The Data Connectivity Service is already assigned to a
network";
         tailf:dependency ".";
        }
      }

      leaf status {
        description "Network status.";
        tailf:info "Status.";
        type identityref {
          base "vts-ids:entity-status";
        }
        mandatory true;
      }
      leaf network-creation-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when the network was triggered";
        tailf:info  "The timestamp when the network was triggered";
      }
      leaf vni-pool {
        description "Vni pool name.";
        tailf:info "Vni pool name.";
        type vts-types:string128;
        // TODO: consider using a leafref to the pool
        default "vnipool";
      }
      uses VXLAN-NETWORK-ID {
        refine "vxlan-network-identifier/user-assigned/vni" {
          description "Vni to use for this network.";
          tailf:info "Vni for the network.";
        }
        refine "vxlan-network-identifier/vts-allocated/vts-allocated-vni" {
```

```
        description "Vni allocated by VTC for this network. The
                     allocation can be dynamic or statically pre-allocated
                     by the user.";
      }
    }
    uses MULTICAST-ADDRESS-IDENTIFIER {
      refine "multicast-ip-address/user-assigned/multicast-address" {
        description "Multicast ip address to use for this network.";
        tailf:info "Multicast ip address for the network.";
      }
      refine "multicast-ip-address/vts-allocated/vts-allocated-multicast" {
        description "Multicast ip address allocated by VTC for this network. The
                     allocation can be dynamic or statically pre-allocated
                     by the user.";
      }
    }
    leaf arp-suppression{
      when "../vts:router-external = 'false'";
      description "Enables BGP EVPN ARP Suppression.";
      tailf:info "ARP suppression.";
      type identityref {
        base "vts-ids:nwk-arp-suppression-type";
      }
      default "vts-ids:nwk-arp-suppression-disabled";
    }
    container vmms {
      list vmm {
        key vmm-id;
        leaf vmm-id {
          description "VMM which has this network.";
          tailf:info "VMM which has this network.";
          type leafref {
            path "/vts:cisco-vts/vts:vmms/vts:vmm/vts:id";
          }
        }
      }
    }
    leaf vts-allocated-evi {
      description "Evi id to use for this virtual entity";
      type vts-types:evi-id;
    }
  } // network
 } // networks
} // grouping VTS-NETWORKS

grouping VTS-BRIDGE-DOMAINS {
 container bridge-domains {
  description "Set of Bridge Domains.";
  tailf:info "Bridge Domains.";
```

```
    list bridge-domain {
      description "List of Bridge Domains.";
      tailf:info "Bridge Domain.";

      key id;

      leaf id {
        description "Network identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }

    } // bridge-domain
  } // bridge-domains
} // grouping VTS-BRIDGE-DOMAINS

grouping VTS-ROUTER-INTERFACES {
  container interfaces {
    list interface {
      description "List of (router) interfaces.";
      tailf:info "Router interfaces.";
      ncs:servicepoint vts-router-interface-servicepoint;
      uses ncs:service-data;

      key subnet-id;

      leaf subnet-id {
        description "Subnetwork identifier.";
        tailf:info "Subnetwork ID.";
        type vts-types:uuid;
        mandatory "true";
      }

      leaf router-id {
        description "Router identifier.";
        tailf:info "Router ID.";
        type vts-types:uuid;
        mandatory "true";
      }
      leaf ip-address {
        description "Router interface IP address.";
        tailf:info "IP address.";
        type inet:ip-address;
      }
      leaf port-edit-timestamp {
          type ietf-yang:date-and-time;
          description "The timestamp when a port in this interface was edited last.";
          tailf:info  "The timestamp when a port in this interface was edited last.";
      }
```

```
      leaf router-gateway-set {
        description "Router gateway set operation";
        tailf:info "Router gateway set operation";
        type identityref {
          base "router-gateway-creation";
        }
        default "unset";
      }
      leaf router-gateway-ip-address {
        description "Router gateway interface IP address.";
        tailf:info "IP address.";
        type vts-types:ip-prefix;
      }
      leaf routerinterface-creation-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when the router interface was triggered";
        tailf:info  "The timestamp when the router interface was triggered";
      }

    } // list interfaces
  } // container interfaces
} // grouping VTS-ROUTER-INTERFACES

grouping TEMPLATE-MAPS {
  list template-map {
    description "List of VTS configuration template applied.";
    tailf:info "VTS Configuration Template Applied.";

    key "template-type";

    leaf template-type {
      description "Template type.";
      tailf:info "Template Type.";
      type vts-types:string128 {
        pattern "route";
      }
    }

    leaf template-name {
      description "Template name.";
      tailf:info "Template Name.";
      mandatory "true";
      type leafref  {
        path "/templates:templates/templates:template/templates:name";
      }
    }

    leaf last-changed {
      description "A timestamp indicating when template configuration applied on
device has changed.
```

```
                This is used to trigger applying of configuration to devices";
      tailf:info "A timestamp indicating when template configuration applied on
device has changed. ";
      type ietf-yang:date-and-time;
     }
   } // template-map
  } // grouping TEMPLATE-MAPS

  grouping VTS-ROUTERS {
   container routers {
    description "Set of routers.";
    tailf:info "Routers.";

    list router {
     description "List of routers.";
     tailf:info "Router.";
     uses ncs:service-data;
     ncs:servicepoint vts-router-servicepoint;

     key id;
     unique "name";
     unique "vrf-name";
     leaf id {
      description "Router identifier.";
      tailf:info "Router ID.";
      type vts-types:uuid;
     }
     leaf name {
      description "Router name.";
      tailf:info "Name.";
      type vts-types:string15;
     }
     leaf vrf-name {
      description "VRF name of the router";
      tailf:info "VRF name.";
      type vts-types:alphanumeric24;
     }
     leaf router-gateway {
      description "Router gateway.";
      tailf:info "Router gateway.";
      type vts-types:uuid;
     }
     leaf router-gateway-ip-address {
      when "../router-gateway";
      description "Router gateway interface IP address.";
      tailf:info "IP address.";
      type string {
          pattern '([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.(([0-9]|[1-9][0-
9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){2}([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-
5])/(([1-9])|([1-2][0-9])|(3[0-2]))' {
```

```
            error-message "Invalid format or invalid ip for router gateway ip address";
          }
        }
      }
      leaf router-gateway-ipv6-address {
        when "../router-gateway";
        description "Router gateway interface IPv6 address.";
        tailf:info "IPv6 address.";
        type vts-types:ipv6-prefix;
      }
      leaf provider-router {
        description "Provider router.";
        tailf:info "Provider router.";
        type boolean;
        default false;
      }

      leaf status {
        description "Router status.";
        tailf:info "Router status.";
        type identityref {
          base "vts-ids:entity-status";
        }
        mandatory true;
      }
      leaf router-creation-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when the router was triggered";
        tailf:info  "The timestamp when the router was triggered";
      }

      uses VXLAN-NETWORK-ID {
        refine "vxlan-network-identifier/user-assigned/vni" {
          description "Vni to use for this router";
          tailf:info "Vni for the network.";
        }
        refine "vxlan-network-identifier/vts-allocated/vts-allocated-vni" {
          description "Vni allocated by VTC for this router The
                       allocation can be dynamic or statically pre-allocated
                       by the user.";
        }
      } // uses VXLAN-NETWORK-ID

      leaf overlay-router-vpn-id {
        description "Identifies which Global Route Leaking Service is associated with
Router";
        tailf:info "Identifies which Global Route Leaking Service is associated with
Router";
        type leafref {
          path "/cisco-vts/vpns/vpn/id";
```

```
        }
        must "/vts:cisco-vts/vts:vpns/vts:vpn[id=current()]/vts:type='vts:vpn-service-
type-internet'" {
          error-message "Overlay-router-vpn-id is pointing to a service that is not of
type internet";
          tailf:dependency "/cisco-vts/vpns/vpn";
        }
      }

      container template-maps {
        description "Container for VTS configuration template to router mappings.";
        tailf:info "VTS Configuration template-router mappings.";
        uses TEMPLATE-MAPS;
      } // container template-maps

      container vmms {
        list vmm {
          key vmm-id;
          leaf vmm-id {
            description "VMM which has this router.";
            tailf:info "VMM which has this router.";
            type leafref {
              path "/vts:cisco-vts/vts:vmms/vts:vmm/vts:id";
            }
          }
        } // list vmm
      } // container vmms

      leaf enable-default-route {
        description "Set default route.";
        tailf:info "Set default route.";
        type boolean;
        default true;
      }
    } // list router
  } // container routers
 } // grouping VTS-ROUTERS

 ////////////////////////////////////////////////////////////////////////////////////////////////////
 //    Template Target Association Types and Groupings definitions
 ////////////////////////////////////////////////////////////////////////////////////////////////////

//target types
 identity target-type {
   description "Base identity for template association target from which others are
derived.";
 }

 identity tenant-target {
   base "target-type";
```

```
  description "Tenant as template association target.";
  tailf:info "Tenant as template association target.";
}

identity router-target {
  base "target-type";
  description "Router as template association target.";
  tailf:info "Router as template association target.";
}

identity network-target {
  base "target-type";
  description "Network as template association target.";
  tailf:info "Network as template association target.";
}

identity device-target {
    base "target-type";
    description "Device as template association target.";
    tailf:info "Device as template association target.";
}

grouping TEMPLATES-DEVICE-GROUP {
  container template-device-groups {
    description "List of device groups that belong to templates";
    tailf:info "List of device groups that belong to templates";

    config true;

    list template-device-group {
      description "Set of devices";
      tailf:info "Set of devices";

      uses ncs:service-data;
      ncs:servicepoint template-device-group-servicepoint;

      key device-group-name;

      leaf device-group-name {
        description "Name of the device-group";
        tailf:info "Name of the device-group";
        type vts-types:non-empty-string255;
      }

      container devices {
        description "List of device in the device group";
        tailf:info "List of device in the device group";

        list device {
          description "device in the device group";
```

```
          tailf:info "device in the device group";
          key name;

          leaf name {
             description "Name of the device";
             tailf:info "Name of the device";
             type vts-types:non-empty-string128;
             must "(/vts:cisco-vts/vts:template-device-groups/vts:template-device-
group[device-group-name='ALL'] and current()='*') or
             (/vts:cisco-vts/vts:template-device-groups/vts:template-device-
group[device-group-name != 'ALL'] and /ncs:devices/ncs:device[ncs:name =
current()])"{
                error-message "deleting device not allowed while devices are still
associated with templates OR
                               device name is not in ncs:devices/ncs:device list  OR
                               device group ALL should have device name as * ";
                tailf:dependency "/ncs:devices/ncs:device/ncs:name";
                tailf:dependency "/vts:cisco-vts/vts:template-device-groups/vts:template-
device-group/vts:device-group-name";
                tailf:dependency ".";
             }
          }
        } //list device
       } // container devices
     } // list template-device-group
   } // container template-device-groups
  } // grouping TEMPLATES-DEVICE-GROUP
  grouping TEMPLATE-TARGET-MAPS {
    container template-target-maps{

     tailf:info "Repository of VTS templates and associated targets.";

     config true;

     list template-target-map {
       description "List of VTS templates and targets associated with.";
       tailf:info "List of VTS templates and targets associated with.";

       uses ncs:service-data;
       ncs:servicepoint template-target-maps-servicepoint;

       key "template-target-id template-type template-name";

       leaf template-target-id{
         tailf:info "Target being path to tenant like /vts:cisco-vts/tenants/tenant{<name
of tenant>} or
         target being path to router like
         /vts:cisco-vts/tenants/tenant{<name of tenant>}/topologies/topology{<name of
topology>}/routers/router{<id of the router>} or
         target being path to network like
```

```
        /vts:cisco-vts/tenants/tenant{<name of tenant>}/topologies/topology{<name of
topology>}/networks/network{<id of the network>}";
        type vts-types:non-empty-string-max;
        //condition for underlay template
        must "((current() = 'na') or ../template-target-type != 'vts:device-target')"{
          error-message "target id should be na for device templates";
          tailf:dependency "../type";
          tailf:dependency ".";
        }

      }

      leaf template-target-type{
        tailf:info "Target like tenant, router,network, device to which the template gets
associated";
        type identityref {
          base "target-type";
        }
        /*
         * Only tenant-target type is validated here. The router-target is validated as
part of vtsRouterRFS postModification
         * For device-target type the check is performed for template-target-id
         * TODO: performance evaluate inclusion of route id validation as part of
must condition and include it here
         */

         must "(current() = 'vts:tenant-target' and
                (count(/vts:cisco-vts/vts:tenants/vts:tenant[vts:name = substring-
before(substring-after(current()/../template-target-id,\"{\"),\"}\")]) = 1)) or
              (current() = 'vts:router-target') or
              (current() = 'vts:network-target') or
              (current() = 'vts:device-target')"{
           error-message "deleting tenant not allowed while templates are still associated
OR
                          template-target-id passed is not valid for tenant-target type  OR
                          template-target-type is not tenant-target or router-target  or network-
target";
           tailf:dependency "/vts:cisco-vts/vts:tenants/vts:tenant/vts:name";
           tailf:dependency ".";
         }
        mandatory true;
      }

      /*TODO:
         1. removed the must condition to check template name and template type
matching as we changed the template
            to pattern due to NCS issue with using a leaf that is identity-ref as a key of
list.
         2. The REST POST call doesn't work while passing the value for the identity-
ref.
```

```
        3. Once that issue is resolved we can make the change in this model hence we
are using qualified names for
           that reason in the pattern
        4. The pattern being on a single line is required as I didn't find a way to
mention that it is continuation
           of OR condition on the new line. Once we have new line the pattern was not
getting matched.
      */

      leaf template-type {
        description "Template type";
        tailf:info "Template Type
                  cisco-vts-templates:l2-extension or
                  cisco-vts-templates:l3-service-extension or
                  cisco-vts-templates:external-gateway-service-extension or
                  cisco-vts-templates:bgp-service-extension or
                  cisco-vts-templates:route or
                  cisco-vts-templates:device";
        type vts-types:string128 {
          pattern 'cisco-vts-templates:external-gateway-service-extension|cisco-vts-
templates:l2-extension|cisco-vts-templates:l3-service-extension|cisco-vts-
templates:bgp-service-extension|cisco-vts-templates:route|cisco-vts-templates:device';
        }
      }

      leaf template-name {
        description "Template name.";
        tailf:info "Template Name.";
        mandatory true;
        type leafref  {
          path "/templates:templates/templates:template/templates:name";
        }
      }

      leaf last-changed {
        description "A timestamp indicating when template configuration applied on
device has changed.
                   This is used to trigger applying of configuration to devices";
        tailf:info "A timestamp indicating when template configuration applied on
device has changed. ";
        type ietf-yang:date-and-time;
      }

      leaf device-group-id{
        tailf:info "device group name referring the group of devices selected by the user
                    for this template-target association";
        type leafref{
          path "../../../template-device-groups/template-device-group/device-group-
name";
        }
```

```
      }
    } // list template-target-map
  } // container templates-target-map
} // grouping TEMPLATE-TARGET-MAPS

container cisco-vts {
  description "cisco-vts root container.
               This serves as a root for all cisco-vts objects.";
  tailf:info "cisco-vts root container.";


  uses SYSTEM-CONFIG;
  uses VTS-INVENTORY;
  // TODO: enforce in backend code that a device can be in only one vertical in the
infra-policy
  uses INFRA-POLICY;
  uses UUID-SERVER;
  uses VTS-DEVICE-GROUP;
  uses VTS-DEVICE-INTERFACE-GROUP;
  uses VTS-NETWORK-TO-PORTS-MAPPING;
  uses TEMPLATES-DEVICE-GROUP;
  uses TEMPLATE-TARGET-MAPS;
  uses VTS-VPNS;
  uses AUTHENTICATION-CONFIG;
  uses VTS-SERVICE-ABSTRACTION;
  uses MVMM-POLICY;
  uses VTS-SFD;

  container tenants {
    description "Set of VTS tenants.";
    tailf:info "Set of VTS tenants.";

    list tenant {
      description "List of VTS tenants.";
      tailf:info "Tenants.";

      key name;
      leaf name {
        description "Tenant name.";
        tailf:info "Name.";
        type vts-types:string15;
      }

      container template-maps {
        presence "Need to invoke template map service point";
        description "Container for VTS configuration template to tenant mappings.";
        tailf:info "VTS Configuration template-tenant mappings.";
        uses ncs:service-data;
        ncs:servicepoint template-tenant-map-servicepoint;
```

```
    uses TEMPLATE-MAPS;

  } //template-maps

  container endpoint-groups {
    uses VTS-EPG;
  }

  container contracts {
    uses VTS-CONTRACT;
  }

  uses VTS-SFI;

  uses COMMON-VTS-TENANT-GROUPING;
  uses POLICY-CONTAINER;

  container network-extensions {
    presence "Need to invoke network extension service point.";
    description "Container to store properties extensible by network and to invoke
network extension service.";
    tailf:info "Container to store properties extensible by network and to invoke
network extension service.";
    uses ncs:service-data;
    ncs:servicepoint network-extensions-servicepoint;
    leaf extend-networks-to-core {
      description "Extend all inherited networks to core.";
      tailf:info "Extend all inherited networks to core.";
      type vts-types:yes-no-none-enum;
      default no;
    }
  }

  container topologies {
    description "Set of topologies belonging to a tenant.";
    //Todo: Add an example for topologies
    tailf:info "Set of topologies belonging to a tenant.";

    list topology {
      description "List of topologies.";
      tailf:info "List of topologies.";

      key id;
      leaf id {
        description "Topology name.";
        tailf:info "Topology name.";
        type vts-types:string128;
      }
      uses vts:VTS-BRIDGE-DOMAINS;
      uses vts:VTS-NETWORKS;
```

```
        uses vts:VTS-SUBNETWORKS;
        uses vts:VTS-PORTS;
        uses vts:VTS-ROUTERS;
        uses STATIC-ROUTES;
        uses vts:VTS-ROUTER-INTERFACES;
      } // list topology
    } // container topologies
  } //list tenants
} // container tenant

uses vts:VTS-SHARED-NETWORKS;
uses vts:VTS-SHARED-SUBNETS;
uses vts:VTS-METADATA;

container vmms {

  grouping IP-PORT-PROTOCOL {
    leaf ip-address {
      description "IP address for VMM.";
      tailf:info "IP address for VMM";
      type inet:ip-address;
    }
    leaf port {
      description "Port used for calling REST APIs of VMM.";
      tailf:info "Port used for calling REST APIs of VMM";
      type uint32;
      //default 35357;
    }
    leaf protocol {
      description "Protocol used for calling REST APIs of VMM";
      tailf:info "Protocol used for calling  REST APIs of VMM";
      type identityref {
        base "rest-protocol";
      }
      //default https;
    }
  } // IP-PORT-PROTOCOL

  grouping COMMON-VMM-REGISTRATION_DETAILS {
    container api-endpoint {
      uses IP-PORT-PROTOCOL;
      leaf userid {
        description "VMM SSH user id";
        tailf:info "VMM SSH user id";
        type vts-types:string128;
        when "../../../vmm/type = 'vts-ids:vcenter'";
      }

      leaf password {
        description "VMM SSH user passphrase.";
```

```
      tailf:info "VMM SSH user passphrase";
      type vts-types:string128;
      when "../../../vmm/type = 'vts-ids:vcenter'";
    }
  }
} //COMMON-VMM-REGISTRATION_DETAILS

grouping OPENSTACK-VMM-REGISTRATION_DETAILS {

  container keystone {

    uses IP-PORT-PROTOCOL;

    leaf tenant-name {
      description "Tenant name.";
      tailf:info "Name.";
      type vts-types:string15;
    }

    leaf tenant-userid {
       description "VMM tenant user id";
       tailf:info "VMM tenant user id";
       type vts-types:string128;
    }

    leaf tenant-password-phrase {
        description "VMM tenant user password.";
        tailf:info "VMM tenant user password";
        type vts-types:string128;
    }

    leaf domain-id {
      description "domain id.";
      tailf:info "domain id";
      type vts-types:string128;
    }

    when "../../vmm/type = 'vts-ids:openstack'";
  } //keystone

  container osp-director {
    leaf userid {
      description "OSP Director SSH user id for OSPD Openstack VMM.";
      tailf:info "OSP Director SSH user id for OSPD Openstack VMM.";
      type vts-types:string128;
    }

    leaf password {
      description "OSP Director SSH user passphrase for OSPD Openstack VMM.";
      tailf:info "OSP Director SSH user passphrase for OSPD Openstack VMM.";
```

```
      type vts-types:string128;
    }

    leaf ip-address {
      description "IP address of OSP Director for OSPD Openstack VMM.";
      tailf:info "IP address of OSP Director for OSPD Openstack VMM";
      type inet:ip-address;
    }
    when "../../vmm/type = 'vts-ids:openstack'";
  } //osp-director
} //OPENSTACK-VMM-REGISTRATION_DETAILS

description "Set of VMM instances registered with VTC.";
list vmm {
  description "List of VMM instances registered with VTC.
               Either standalone or clusters.";
  tailf:info "VMM instance.";

  key id;
  unique api-endpoint/ip-address;
  unique osp-director/ip-address;

  leaf id {
    description "VMM identifier.";
    tailf:info "ID.";
    type vts-types:uuid;
  }

  leaf type {
    description "Type of vmm.";
    tailf:info "Type of vmm.";
    type identityref {
      base "vts-ids:vmm-type";
    }
    mandatory true;
  }


  leaf install-vmm-components{
    description "Determines if VTS Installs VMM components or not";
    tailf:info  "Determines if VTS installs VMM components or not";
    type boolean;
    default true;
  }

  leaf version {
    description "vmm version.";
    tailf:info "vmm version.";
    type identityref {
      base "vts-ids:vmm-version";
```

```
        }
        must "starts-with(current(), current()/../type)" {
          error-message "VMM version must match the VMM type.";
          tailf:dependency "../type";
        }
        mandatory true;
      }

      leaf status {
        description "Registration status.";
        tailf:info "Registration status.";
        type identityref {
          base "vts-ids:vmm-registration-state";
        }
        default vts-ids:vmm-registration-initial;
      }

      leaf message {
        description "Detailed message about current registration state.";
        tailf:info "Detailed message about current registration state.";
        type vts-types:string255;
      }

      leaf datacenter {
          description "Datacenter for vcenter.";
          tailf:info "Datacenter for vcenter.";
          type vts-types:string128;
          when "../type = 'vts-ids:vcenter'";
      }

      list controller-nodes {
        description "List of controller nodes";
        tailf:info "List of controller nodes";
        key id;
        unique ip-address;

        leaf id {
          description "controller node identifier.";
          tailf:info "controller node identifier.";
          type vts-types:uuid;
        }

        leaf ip-address {
          description "IP addresses of the VMM. For standalone VMMs, a single IP
address will be provided.
                       For clustered VMMs, each IP address correspond to a cluster node.";
          tailf:info "Controller/Network node ip address";
          type inet:ip-address;
          must "boolean(../../osp-director/ip-address) or (count(../../../vmm/controller-
nodes[ip-address=current()]) = 1)" {
```

```
            error-message "The provided controller IP is already associated with
another VMM";
            tailf:dependency "../../osp-director/ip-address";
            tailf:dependency ".";
          }
          //min-elements 1;
          //must "../type = 'vts-ids:openstack' or (../type = 'vts-ids:vcenter' and
count(../ip-address) = 1)" {
          //  error-message "Only 1 ip-address can be specified for vcenter";
          //  tailf:dependency "../type";
          //}
          //must "count(../../vmm[ip-address=current()])=1" {
          //  error-message "The provided ip-address is already assigned to a VMM.";
          // tailf:dependency ".";
          //}

        } // Question we should remove the above must conditions and min-elements?

        leaf port {
          description "host port.";
          tailf:info "host port.";
          type uint32;
          default 443;
          when "../../type = 'vts-ids:vcenter'";
        }

        leaf userid {
          description "VMM SSH user id";
          tailf:info "VMM SSH user id";
          type vts-types:string128;
        }

        leaf password {
          description "VMM SSH user passphrase.";
          tailf:info "VMM SSH user passphrase";
          type vts-types:string128;
        }

        leaf status {
          description "Installation status.";
          tailf:info "Installation status.";
          type identityref {
            base "vts-ids:vmm-registration-state";
          }
          default vts-ids:vmm-registration-initial;
        }

        leaf message {
          description "Detailed message about current installation state.";
          tailf:info "Detailed message about current installation state.";
```

```
          type vts-types:string255;
        }
      }

      leaf vtf-deployment {
        description "Deployment mode for the VTFs.";
        tailf:info "Deployment mode for the VTFs.";
        type identityref {
          base "vts-ids:vtf-deployment-mode";
        }
        when "../type = 'vts-ids:openstack' and (../version = 'vts-ids:openstack-liberty-
centos' or ../version ='vts-ids:openstack-liberty-rhel')";
      }

      leaf name {
        description "A name of the VMM";
        tailf:info "A name of the VMM";
        type vts-types:non-empty-string128;
        must "not(current() = 'VTS') and not(current() = 'Vts')
        and not(current() = 'vts') and not(current() = 'VTs')
        and not(current() = 'vTs') and not(current() = 'vtS')
        and not(current() = 'vTS') and not(current() = 'VtS')" {
          error-message "VMM name cannot be VTS.";
          tailf:dependency ".";
        }
        mandatory true;
      }

      leaf mode {
        description "Indicates mode of VMM";
        tailf:info "Indicates mode of VMM";
        type identityref {
          base "vmm-mode";
        }
        default trusted;
      }

      uses COMMON-VMM-REGISTRATION_DETAILS;
      uses OPENSTACK-VMM-REGISTRATION_DETAILS;

    } //vmm
  } //vmms
  container trunks {
    description "Set of trunks.";
    tailf:info "Trunks.";
    list trunk {
      description "List of trunks";
      tailf:info "Trunk.";
      uses ncs:service-data;
      ncs:servicepoint vts-trunk-servicepoint;
```

```
      key id;

      leaf id {
        description "Trunk identifier";
        tailf:info "Trunk ID.";
        type vts-types:uuid;
      }
      uses vts:COMMON-TRUNK;

      leaf tenant-name {
        description "Tenant derived from parent port.";
        tailf:info "Tenant derived from parent port.";
        type leafref {
          path "../../../tenants/tenant/name";
        }
      }
      leaf topology-id {
        description "Topology derived from parent port.";
        tailf:info "Topology derived from parent port.";
        type leafref {
          path "../../../tenants/tenant/topologies/topology/id";
        }
      }
    }
  }
 } //container cisco-vts
}
```

6.4 VTS Common

```
submodule cisco-vts-common {

 belongs-to cisco-vts {
  prefix vts;
 }

 import ietf-inet-types {
  prefix inet;
 }
 import ietf-yang-types {
  prefix yang;
 }

 import tailf-common {
  prefix tailf;
 }
 import tailf-ncs {
```

```
  prefix ncs;
 }
 import cisco-vts-types {
  prefix vts-types;
 }
 import cisco-vts-identities {
  prefix vts-ids;
 }

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
   for Cisco's VTS.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

 revision "2017-04-20" {
    description
      "VTS 2.6
       Enhancements:
         - Added leaf virtual-switch to the grouping COMMON-SERVER.
         - Added leaf nodes sriov-enabled and physnet-name to the grouping
COMMON-SERVER.
      ";
 }
 revision "2016-07-15" {
  description
    "VTS 2.3.0
     Enhancements:
       - Added leaves to grouping COMMON-SUBNETWORK
           - IPv6-related fields: ipv6-ra-mode, ipv6-address-mode
           - subnetpool-id
       - Added COMMON-ESI-GROUPING grouping for ESI nodes
     Corrections:
       - Modified cidr type from
           tailf:ipv4-address-and-prefix-length to
           tailf:ip-address-and-prefix-length
   ";
```

```
}

revision "2015-02-28" {
 description
    "Initial revision.";
}

/*
 * Groupings
 */
grouping COMMON-NETWORK {
  description "Set of common data nodes for virtual networks.";
  leaf admin-state-up {
    description "The network administrative state is UP.";
    tailf:info "Administrative state up.";
    type boolean;
    default "true";
  }
  leaf name {
    description "Network name.";
    tailf:info "Name.";
   type vts-types:string128;
  }
  leaf provider-physical-network {
    description "Physical network.";
    tailf:info "Physical network.";
   type vts-types:string128;
  }
  leaf provider-segmentation-id {
    description "Segmentation identifier.";
    tailf:info "Segmentation ID.";
   type vts-types:string128;
  }
  leaf provider-network-type {
    description "Network type.";
    tailf:info "Network type.";
    type identityref {
      base "vts-ids:network-type";
    }
  }
  leaf router-external {
    description "External network.";
    tailf:info "External network.";
    type boolean;
    default "false";
  }
  leaf shared {
    description "Shared network.";
    tailf:info "Shared.";
    type boolean;
```

```
    default "false";
  }
}

grouping COMMON-SUBNETWORK {
  description "Set of common data nodes for virtual subnetworks.";
  leaf name {
    description "Subnetwork name.";
    tailf:info "Name.";
    type vts-types:string128;
  }
  leaf cidr {
    description "Subnetwork address.";
    tailf:info "Subnetwork address.";
    type tailf:ip-address-and-prefix-length;
    mandatory true;
  }
  leaf enable-dhcp {
    description "Enable DHCP.";
    tailf:info "Enable DHCP.";
    type boolean;
    default "false";
  }
  leaf gateway-ip {
    description "Gateway IP address.";
    tailf:info "Gateway IP address.";
    type inet:ip-address;
    mandatory true;
  }
  leaf ip-version {
    description "IP version.";
    tailf:info "IP version.";
    type enumeration {
      enum 4;
      enum 6;
    }
    default 4;
  }
  leaf shared {
    description "Shared subnetwork.";
    tailf:info "Shared.";
    type boolean;
    default "false";
  }
  leaf subnetpool-id {
    description "Identifier for the subnetpool from where to obtain subnet address.";
    tailf:info "Subnetpool identifier.";
    type vts-types:uuid;
  }
```

```
  // IPv6-related fields
  leaf ipv6-ra-mode {
    description "Determines how router advertisement in handled.";
    tailf:info "Router adverstisement handled.";
    type identityref {
      base "vts-ids:ipv6-address-assignment";
    }
  }
  leaf ipv6-address-mode {
    description "Determines how addressing in handled.";
    tailf:info "Addressing handling.";
    type identityref {
      base "vts-ids:ipv6-address-assignment";
    }
  }
}

grouping COMMON-PORT {
  description "Set of common data nodes for virtual ports.";
  leaf name {
    description "Port name.";
    tailf:info "Name.";
   type vts-types:string128;
  }
  leaf admin-state-up {
    description "The network administrative state is UP.";
    tailf:info "Administrative state up.";
    type boolean;
    default "true";
  }
  leaf mac-address {
    description "Port MAC address.";
    tailf:info "MAC address.";
    type union {
      type yang:mac-address;
      type vts-types:unknown-mac;
    }
    mandatory true;
  }
  leaf device-id {
    description "Device Identifier.";
    tailf:info "Device Identifier.";
    type vts-types:string128;
  }
  leaf device-owner {
    description "Device owner.";
    tailf:info "Device owner.";
   type vts-types:string128;
  }
  leaf binding-host-id {
```

```
      description "Host identifier";
      tailf:info "Host identifier";
      type vts-types:string128;
    }
    leaf physnet-name {
      description "Name of the physnet associated with physical interface mentioned in
interface-name";
      tailf:info "Name of the physnet associated with physical interface mentioned in
interface-name";
      type vts-types:string128;
    }
    // NOTE: security-groups subtree may move in the future.
    //       It is an "extra" in our driver that is not part of
    //       the ML2 driver API. If ML2 adds official support we
    //       likely move this tree to match.
    // TODO: WHAT TO DO W/ SECURITY GROUPS
    list security-groups {
      description "List of security groups.";
      tailf:info "Security group.";
      key id;
      leaf id {
        description "Security group identifier.";
        tailf:info "Identifier.";
        type vts-types:uuid;
      }
      leaf name {
        description "Security group name.";
        tailf:info "Name.";
        type vts-types:string128;
      }
      leaf tenant-id {
        //TODO: do we need this? the leafref to the network may suffice
        description "Tenant identifier.";
        tailf:info "Tenant ID.";
        type vts-types:string128;
        // TODO: consider moving to leafref
        //type leafref {
        //  path "/tenant:tenant/tenant:tenant-info/" +
        //       "tenant:vmm-tenant-id";
        //}
      }
      leaf tenant-name {
        // TODO: do we need to name too? The tenant-id may suffice
      description "Tenant name.";
      tailf:info "Tenant name.";
       type vts-types:string128;
      }
      leaf description {
        description "Security group description.";
        tailf:info "Description.";
```

```
  type vts-types:string128;
 }
 list security-group-rules {
  description "List of security group rules.";
  tailf:info "Security group rule.";
  key id;
  leaf id {
   description "Security group rules identifier.";
   tailf:info "ID.";
   type vts-types:uuid;
  }
  leaf remote-group-id {
   description "Remote group identifier.";
   tailf:info "Remote group ID.";
   type vts-types:string128;
  }
  leaf direction {
   description "Direction.";
   tailf:info "Direction.";
   type vts-types:string128;
   // TODO: reconsider as an identity
  }
  leaf remote-ip-prefix {
   description "Remote IP address prefix.";
   tailf:info "Remote IP address prefix.";
   type inet:ip-prefix;
  }
  leaf protocol {
   description "Protocol.";
   tailf:info "Protocol.";
   type vts-types:string128;
   // TODO: reconsider as an identity
  }
  leaf ethertype {
   description "Ethertype.";
   tailf:info "Ethertype.";
   type vts-types:string128;
   // TODO: reconsider as an identity
  }
  leaf port-range-max {
   description "Upper bound of the port range.";
   tailf:info "Port range (maximum).";
   type uint32;
  }
  leaf port-range-min {
   description "Lower bound of the port range.";
   tailf:info "Port range (minimum).";
   type uint32;
   must "../port-range-min <= ../port-range-max" {
    error-message "The minimum in the port range cannot be " +
```

```
                    "greater than the maximum";
          tailf:dependency "../port-range-max";
        }
      }
      leaf tenant-id {
        description "Tenant identifier.";
        tailf:info "Tenant ID.";
        type vts-types:string128;
      }
      leaf security-group-id {
        description "Security group indentifier.";
        tailf:info "Security group ID.";
       // TODO: reconsider a maximum length for this leaf.
       type string;
      }
    }
  }
}

grouping COMMON-PHYSICAL-PORT {
  description "Set of common data nodes for physical ports.";
  leaf name {
    description "Port name.";
    tailf:info "Name.";
    type vts-types:string128;
  }
  leaf local-mac {
    description "Mac address of the port.";
    tailf:info "Mac address.";
    type yang:mac-address;
  }
  leaf connection-type {
    description "The type of the connection. That is, to " +
                "what type of device this port is connected " +
                "to.";
    tailf:info "Type of device this port is connected to.";
    type identityref {
      base "vts-ids:peer-type";
    }
  }
  leaf mode {
    description "Mode of the port.";
    tailf:info "Port mode.";
    type identityref {
      base "vts-ids:network-layer";
    }
  }
  leaf group-name {
    description "Group Name.";
    tailf:info "Group Name.";
```

```
      type vts-types:string128;
    }
  }

  // Device grouping
  grouping VTS-DEVICE-METADATA {
     leaf group-tag {
       description "Group Tagging of Device";
       tailf:info "Group Tag";
       type vts-types:string128;
     }
  }
  // Server groupings
  grouping COMMON-SERVER {
    description "Set of common data nodes for servers.";
    leaf name {
       description "Name of the server (Hostname typically).";
       tailf:info "Server name.";
       must "not(../../../connection-type = 'vts-ids:fabric') or " +
        " (../../../connection-type = 'vts-ids:fabric' and  /ncs:devices/ncs:device[ncs:name
= current()])"{
          error-message "In fabric connection-type, valid device should be present.
          Else connection-type should be server or empty.";
          tailf:dependency "../../../connection-type";
        }
       type vts-types:string128;
    }
    leaf type {
      description "Type of server.";
      tailf:info "Type of server.";
      type identityref {
        base "vts-ids:server-type";
      }
      must "not(current()='vts-ids:virtual-server') or current()/../ip" {
             error-message "For virtual servers and VTFs an ip address must be
provided.";
      }
    }
    leaf mac {
      description "Mac address of the server interface " +
                  "connected to this switch port.";
      tailf:info "Mac address of the server interface " +
                  "connected to this switch port.";
      type yang:mac-address;
    }
    leaf interface-name {
      description "Name of the server interface connected " +
                  "to this switch port.";
      tailf:info "Name of the server interface connected " +
                  "to this switch port.";
```

```
      type vts-types:string128;
    }
    leaf sriov-enabled {
      description "Indicates if SRIOV is enabled or not on this physical interface.";
      tailf:info "Indicates if SRIOV is enabled or not on this physical interface.";
      type boolean;
    }
    leaf physnet-name {
      description "Name of the physnet associated with physical interface mentioned in
interface-name";
      tailf:info "Name of the physnet associated with physical interface mentioned in
interface-name";
      type vts-types:string128;
    }
    leaf ip {
      description "IP address of the server interface " +
                  "connected to this switch port.";
      tailf:info "IP address of the server interface " +
                  "connected to this switch port.";
      type inet:ip-address;
    }
    leaf vmm-id {
      description "VMM which manages the server.";
      tailf:info "VMM which manages the server.";
      type leafref {
        path "/cisco-vts/vmms/vmm/id";
      }
      must "(../type='vts-ids:virtual-server')" {
        error-message "VMM Id can be provided only for virtual-servers.";
      }
    }
    leaf capability {
      description "Capability of the server.";
      tailf:info "Capability of the server. " +
                 "physical or virtual.";
      type identityref {
        base "vts-ids:server-capability";
      }
      must "(../type='vts-ids:virtual-server')" {
        error-message "Capability can be provided only for virtual servers.";
      }
    }

    leaf virtual-switch {
      description "Type of switch.";
      tailf:info "Type of switch.";
      type identityref {
        base "vts-ids:switch-type";
      }
      must "(../type='vts-ids:virtual-server')" {
```

```
      error-message "Virtual Switch can be provided only for virtual servers.";
      tailf:dependency "../type";
     }
     must "(current() = 'vts-ids:not-defined-st') or (current() != 'vts-ids:dvs-st' and
(current()/../vmm-id and /vts:cisco-vts/vts:vmms/vts:vmm[id = current()/../vmm-
id]/vts:type='vts-ids:openstack')) or
          ((current() != 'vts-ids:ovs-st' and current() != 'vts-ids:vtf-l2-st') and
(current()/../vmm-id and /vts:cisco-vts/vts:vmms/vts:vmm[id = current()/../vmm-
id]/vts:type='vts-ids:vcenter'))
          "{
      error-message "The following combination Virtual Switch OVS can be set only
for host managed by Openstack. OR
                  Virtual Switch DVS can be set only for host managed by vCenter. OR
                  Virtual Switch VTF-L2 can be set only for host managed by
Openstack";
      tailf:dependency ".";
     }
   }

   leaf vtf-link {
    description "Type of vtf-connection.";
    tailf:info "Type of vtf-connection.";
    type identityref {
     base "vts-ids:vtf-link";
    }
   }

   leaf username {
    description "Username of server.";
    tailf:info "Username of server.";
    type string;
   }
   leaf password {
    description "Password of server.";
    tailf:info "Password of server.";
    type string;
   }
   leaf install-status {
    description "Status of installation.";
    tailf:info "Status of installation.";
    type identityref {
     base "vts-ids:install-status";
    }
    must "(../type='vts-ids:virtual-server')" {
     error-message "Install-status is applicable only for virtual servers.";
    }
   }
   leaf install-status-message {
    description "Detailed message describing " +
                "status of installation.";
```

```
    tailf:info "Detailed message describing " +
               "status of installation.";
    type vts-types:string255;
    must "(../type='vts-ids:virtual-server')" {
      error-message "Install-status-message is applicable only for virtual servers.";
    }
  }
}

// TODO: If allocators do not go under cisco-vts, move this out
grouping POOL-ALLOCATIONS {
  description "List of allocations from the pool.";
  list allocation {
    description "List of pool allocations.";
    tailf:info "Pool allocation.";
    tailf:cli-suppress-mode;
    key "id";
    leaf id {
      description "Pool allocation identifier.";
      tailf:info "ID.";
      type uint32;
    }
    list owner {
      description "List of owners of the pool allocation.";
      tailf:info "Owner of the pool allocation.";
      key owner_name;
      leaf owner_name {
        description "Name of the owner of the pool allocation.";
        tailf:info "Name.";
        type vts-types:string128;
      }
    }
  }
}

grouping COMMON-VTS-TENANT-GROUPING {
  description "Set of common data nodes for vts tenants.";
  leaf description {
    description "Tenant description.";
    tailf:info "Description.";
    type vts-types:string128;
  }
  // Is this required? Should it be a leafref to uuid of tenant in vmm?
  leaf vmm-tenant-id {
    description "Tenant identifier.";
    tailf:info "ID.";
    type vts-types:string128;
  }

  container vmms {
```

```
    list vmm {
      key vmm-id;
      leaf vmm-id {
        description "VMM which has this tenant.";
        tailf:info "VMM which has this tenant.";
        type leafref {
          path "/vts:cisco-vts/vts:vmms/vts:vmm/vts:id";
        }
      }
    }
  }

 }

 grouping COMMON-ESI-GROUPING {
  description "Set of common ESI nodes.";
  leaf es-id {
    description "Ethernet segment identifier.";
    tailf:info "Ethernet segment identifier.";
    type uint32;
  }
  leaf system-mac {
    description "System Mac address.";
    tailf:info "System Mac address.";
    type string {
      tailf:info "eeee.eeee.eeee;;Specify system mac address (Option 4)";
      pattern "[0-9a-f]{4}(\.[0-9a-f]{4}){2}";
    }
  }
 }


}
```

6.5 UUID Server

```
submodule cisco-vts-uuid-server {

 belongs-to cisco-vts {
  prefix vts;
 }

 import cisco-vts-types {
  prefix vts-types;
 }
 include cisco-vts-common {
  revision-date 2017-04-20;
 }

 import tailf-common {
```

```
  prefix tailf;
 }
 import tailf-ncs {
  prefix ncs;
 }

 include cisco-vts-inventory {
  revision-date 2016-08-30;
 }
 import cisco-vts-identities {
  prefix vts-ids;
 }

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the inventory
   of the physical topology.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

 revision "2017-04-20" {
  description
   "VTS 2.6
    Enhancements:
      - Added leaves virtual-switch and vtf-link to the grouping UUID-SERVER.
      - Added leaf nodes sriov-enabled and physnet-name to the grouping UUID-
SERVER.
    ";
 }

 revision "2016-08-30" {
  description
   "VTS 2.3.1
    Enhancements:

    Corrections:
   - Relaxed must constraint in the leaf interface-name
   ";
```

```
}

revision "2016-07-15" {
  description
    "VTS 2.3.0
     Enhancements:
       - Added esi container for Ethernet segment identifier
     Corrections:
   ";

}

revision "2015-06-23" {
 description
    "Initial revision.";
}

grouping UUID-SERVER {
  container uuid-servers {
    description "Set of UUID servers";
    tailf:info "Set of UUID servers";
    list uuid-server {

      uses ncs:service-data;

      tailf:info "UUID to Server mapping";

      key "connid";

      leaf connid {
        tailf:info "Unique ID for a tor name, tor port, server name and server port";
        type vts-types:uuid;
      }

      leaf vpcid {
        tailf:info "VPC ID for this mapping";
        type vts-types:string128;
        mandatory false;
      }

      container esi {
        description "Ethernet segment identifier for port-channel interface.";
        tailf:info "Ethernet segment identifier.";
        presence "ESI";

        uses vts:COMMON-ESI-GROUPING;
      } // container esi

      leaf torname {
        tailf:info "Tor name";
```

```
        type leafref {
          path "/cisco-vts/devices/device/name";
        }
      }

      leaf portname {
        tailf:info "Port name";
        type vts-types:string512;
        must "/cisco-vts/devices/device/ports/port/name or " +
            "/cisco-vts/devices/device/ports/port/fexs/fex/ports/port/name" {
          error-message "portname should be one of the ports/port/name " +
                      " or ports/port/fexs/fex/ports/port/name";
          tailf:dependency "/cisco-vts/devices/device/ports/port/name";
          tailf:dependency "/cisco-
vts/devices/device/ports/port/fexs/fex/ports/port/name";
        }
      }
      leaf server-id {
        tailf:info "ID of server returned by openstack";
        type vts-types:string512;
        must "/cisco-vts/devices/device/ports/port/servers/server/name or " +
            "/cisco-vts/devices/device/ports/port/fexs/fex/fex-id" {
          error-message "server name should be one of the ports/port/servers/server " +
                      " or ports/port/fexs/fex/name";
          tailf:dependency "/cisco-vts/devices/device/ports/port/servers/server/name";
          tailf:dependency "/cisco-vts/devices/device/ports/port/fexs/fex/fex-id";
        }
      }
      leaf interface-name {
        tailf:info "Port on which the server is attached";
        type vts-types:string512;
        //must "/cisco-vts/devices/device/ports/port/servers/server/interface-name or " +
        //    "/cisco-vts/devices/device/ports/port/fexs/fex/ports/port/name" {
        //  error-message "server interface should be one of the ports/port/servers/server
" +
        //              " or ports/port/fexs/fex/ports/port/name";
        //  tailf:dependency "/cisco-
vts/devices/device/ports/port/servers/server/interface-name";
        //  tailf:dependency "/cisco-
vts/devices/device/ports/port/fexs/fex/ports/port/name";
        //}
      } // interface-name
      leaf server-type {
        description "Type of server.";
        tailf:info "Type of server.";
        type identityref {
          base "vts-ids:server-type";
        }
      }
      leaf sriov-enabled {
```

```
        description "Indicates if SRIOV is enabled or not on this physical interface.";
        tailf:info "Indicates if SRIOV is enabled or not on this physical interface.";
        type boolean;
       }
       leaf physnet-name {
        description "Name of the physnet associated with physical interface mentioned
in interface-name";
        tailf:info "Name of the physnet associated with physical interface mentioned in
interface-name";
        type vts-types:string128;
       }
       leaf virtual-switch {
        description "Type of switch.";
        tailf:info "Type of switch.";
        type identityref {
         base "vts-ids:switch-type";
        }
        default vts-ids:not-defined-st;
       }

       leaf vtf-link {
        must "../server-type='vts-ids:virtual-server'";
        description "Type of vtf-connection.";
        tailf:info "Type of vtf-connection.";
        type identityref {
         base "vts-ids:vtf-link";
        }
       }
       leaf group-id {
        description "Multi Homing group Id";
        tailf:info "Multi Homing group.";
        type vts-types:string128;
       }
     } // list of uuid-server
   }  // container uuid-servers
  } // grouping UUID-SERVER
} // submodule cisco-vts-uuid-server
```

6.6 Infrastructure Policy

```
submodule cisco-vts-infrastructure-policy {

  belongs-to cisco-vts {
    prefix vts;
  }

  import tailf-common {
    prefix tailf;
  }
```

```
import tailf-ncs {
  prefix ncs;
}

import cisco-vts-profiles {
  prefix vts-profiles;
}

import ietf-inet-types {
    prefix inet;
}

import ietf-yang-types {
  prefix ietf-yang;
}

include cisco-vts-common {
  revision-date 2017-04-20;
}
include cisco-vts-inventory {
  revision-date 2016-08-30;
}
import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix vts-ids;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS  management. Specifically, for the
   management of the infrastructure policy.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2017-04-28" {
  description
```

```
    "VTS 2.6
     Enhancements:
       - Added l2-vs-groups to group the host based on the virtual switch type usage.
    ";
  }

  revision "2016-08-30" {
    description
     "VTS 2.3.1
      Enhancements:
        - Added leaf gwgroup-change-timestamp to all the devices which track gateway
group changes
      Corrections:
    - changed dc-gw-parent from choice to presence container
    - changed cisco-vts-profiles prefix from profiles to vts-profiles
     ";

  }

  revision "2016-07-15" {
    description
     "VTS 2.3.0
      Enhancements:

      Corrections:
        - Modified leafref paths from
            /cisco-vts/devices/device/name to
            /ncs:devices/ncs:device/ncs:name

    ";

  }



  revision "2015-06-21" {
    description
     "Initial revision.";
  }

  identity iccp-group-type {
     description "Base identity for iccp groups - either fabric or core";
  }

  identity iccp-fabric-group {
    base iccp-group-type;
    description "Fabric (VXLAN) group type";
    tailf:info "Fabric (VXLAN) group type";
  }
```

```
identity iccp-core-group {
  base iccp-group-type;
  description "Core (MPLS) group type";
  tailf:info "Core  (MPLS) group type";
}


grouping DCI-GROUP-REDUNDANCY-SETTINGS {
  container redundancy-group {
    description "Nodes in this container control the " +
                "Redundancy/Availablity settings of an l3-dci-group.";
    tailf:info "Redundancy/Availability parameters.";

    presence "DCI Group Redundancy";
    uses ncs:service-data;
    ncs:servicepoint dci-group-redundancy-settings-servicepoint;

    container iccp {
      description "Nodes in this container control the " +
                  "redundancy ICCP behavior of an l3-dci-group.";
      tailf:info "ICCP parameters.";

      container groups {
        description "Nodes in this container control the " +
         "redundancy ICCP fabric and core groups of an l3-dci-group.";
        tailf:info "Redundancy ICCP group parameters.";
        list group {
          key id;
          leaf id {
            type uint32 {
              tailf:info "<1-4294967295>;;Enter group number";
              range "1..4294967295";
            }
            mandatory true;
          }
          leaf group-type {
            description "ICCP Group Type (Fabric or Core).";
            tailf:info "ICCP Group Type (Fabric or Core).";
            type identityref {
              base iccp-group-type;
            }
            mandatory true;
          }
        }
      } // container groups
    } // container iccp

    container esi {
      description "Nodes in this container controlthe " +
          "redundancy ESI core-esi of  an l3-dci-group.";
```

```
      tailf:info "ESI core-esi parameter.";

      leaf core-esi {
        type string;
        description "Configure ESI core group number <0-ff>.;;9-octet ESI value";
        tailf:info "Configure ESI core group number <0-ff>.;;9-octet ESI value";
      }
    }

    container anycast {
      description "Nodes in this container control the " +
          "redundancy anycast paramaters of an l3-dci-group.";
      tailf:info "Anycast parameters.";

      leaf loopback-id {
        tailf:info "loopback id number.";
        type uint32 {
          tailf:info "<0-2147483647>";
          range "0..2147483647";
          }
        } // leaf loopback-id
    }  // container anycast

    uses vts-types:MODIFIED-TIME-ELEMENT;
  } // container redundancy-group
}  // grouping DCI-GROUP-REDUNDANCY-SETTINGS

grouping GROUP-POLICY-PARAMS {
  leaf control-plane-protocol {
    description
      "Control plane protocol used to learn end point addresses.";
    tailf:info "Control plane protocol.";
    type identityref {
      base "vts-ids:control-plane-protocol";
    }
  }
  leaf arp-suppression {
    description
      "Enables BGP EVPN ARP suppression.";
     tailf:info "ARP suppression.";
    type empty;
  }
  leaf packet-replication {
    description
      "Packet replication mechanism. It determines how" +
      "packets sent to multiple recipients are " +
      "replicated, so that each recipient gets a copy.";
    tailf:info "Packet replication.";
    type identityref {
      base "vts-ids:packet-replication-mode";
```

```
    } // type identityref
   }  // leaf packet-replication
  } // grouping GROUP-POLICY-PARAMS

  grouping GROUP-POLICY-PARAMS-L3GW {
   container policy-parameters {
    description "Nodes in this container control the " +
                "behavior of an L3 group of " +
                "the infrastructure policy.";
    tailf:info "Policy parameters.";
    leaf distribution-mode {
     description
      "Mode of distribution.";
     tailf:info "Distribution mode.";
     type identityref {
      base "vts-ids:l3-distribution-mode";
     }
    }
    uses GROUP-POLICY-PARAMS {
     refine "packet-replication" {
      must "count(../../../l3-gateway-group/policy-parameters/packet-replication !=
current()/../packet-replication) = 0" {
       error-message "L3 gateway groups within the same administrative " +
                     "domain must use the same packet replication mechanism.";
       tailf:dependency ".";
      }
     } // refine "packet-replication"
    } // uses GROUP-POLICY-PARAMS
   } // container policy-parameters
  } // grouping GROUP-POLICY-PARAMS-L3GW

  grouping INFRA-POLICY {
   container infra-policy {
    container admin-domains {
     description "Set of administrative domains.";
     tailf:info "Administrative domains.";
     list admin-domain {
      description "List of administrative domains.";
      tailf:info "Administrative domain.";
      key name;
      leaf name {
       description "Name of the administrative domain.";
       tailf:info "Name.";
       type vts-types:string128;
      }
      leaf description {
       description "Description of the administrative domain.";
       tailf:info "Description.";
       type vts-types:string128;
      }
```

```
        container l2-vs-groups {
          description "Sets of layer 2 virtual switch groups.";
          tailf:info "Layer 2 virtual switch groups.";
          list l2-vs-group {
            description "List of layer 2 virtual switch group.";
            tailf:info "Layer 2 virtual switch group.";
            key name;
            leaf name {
              description "Name of the L2 virtual switch group.";
              tailf:info "Name.";
              type vts-types:string128;
              must "count(../../../../admin-domain/l2-vs-groups/l2-vs-group[name =
current()/../name]) =1" {
                error-message "The name of a l2 virtual switch group must be unique
system wide.";
                tailf:dependency ".";
              }
            }
            leaf description {
              description "Description of Layer 2 virtual switch group.";
              tailf:info "Description.";
              type vts-types:string128;
            }

            container vtsr-devices {
              list vtsr-device {
                description "List of associated  vtsr devices when group-type is vtf-l2-vs-
group";
                tailf:info "List of associated vtsr devices when group-type is vtf-l2-vs-
group";
                key name;
                leaf name {
                  description "Name of VTSR device that is part of VTS inventory.";
                  tailf:info "Name of VTSR device that is part of VTS inventory.";
                  type leafref  {
                    path "/ncs:devices/ncs:device/ncs:name";
                  }
                }
              } // list vtsr-device
            } //container vtsr-devices

            container hosts {
              description "Set of hosts, acting as layer 2 virtual switches.";
              tailf:info "Hosts in the L2 virtual switch group.";
              list host {
                key "name group-type";
                leaf name {
                  description "Name of the host.";
                  tailf:info "Host Name.";
                  type vts-types:string128;
```

```
            } // leaf name

            leaf group-type {
              description "Type of the L2 virtual switch group this host belongs";
              tailf:info "Type of the L2 virtual switch group this host belongs.";
              type identityref {
                base "vts-ids:l2-vs-group-type";
              }
            }

            leaf vtf_ip {
              when "../vts:group-type = 'vts-ids:vtf-l2-vs-group'";
              description "IP address of the VTF instance on the host, when group-type
is vtf-l2-vs-group";
              tailf:info "IP address of the VTF instance on the host, when group-type is
vtf-l2-vs-group";
              type inet:ip-address;
            } // leaf vtf_ip

            leaf l2-vs-group-change-timestamp {
              type ietf-yang:date-and-time;
              description "The timestamp when the group change was triggered";
              tailf:info  "The timestamp when the group change was triggered";
            } // leaf forwarder-group-change-timestamp
          } // list host
        } // container hosts
        leaf ad-l2-gw-parent {
          description "This group's L2 gateway parent in " +
                      "the current administrative domain " +
                      "in the infrastructure policy hierarchy.";
          tailf:info "L2 gateway in current admin domain.";
          type leafref {
            path "../../../l2-gateway-groups/l2-gateway-group/name";
          }
        }// leaf ad-l2-gw-parent
      }// list l2-vs-group
    }// container l2-vs-groups

    container l2-gateway-groups {
      description "Set of layer 2 gateway groups.";
      tailf:info "Layer 2 gateway groups.";
      list l2-gateway-group {
        description "List of layer 2 gateway groups.";
        tailf:info "Layer 2 gateway group.";
        key name;
        leaf name {
          description "Name of the L2 gateway group.";
          tailf:info "Name.";
          type vts-types:string128;
```

```
            must "count(../../../../admin-domain/l2-gateway-groups/l2-gateway-
group[name = current()/../name]) =1" {
              error-message "The name of a l2 gateway group must be unique system
wide.";
              tailf:dependency ".";
            }
          }
          leaf description {
            description "Description of Layer 2 gateway group.";
            tailf:info "Description.";
            type vts-types:string128;
          }

          container devices {
            description "Set of layer 2 devices, acting as layer 2 gateways.";
            tailf:info "Devices.";
            list device {
              uses ncs:service-data;
              ncs:servicepoint l2-gateway-group-servicepoint;
              key name;
              leaf name {
                description "Name of the device.";
                tailf:info "Name.";
                type leafref {
                  path "/ncs:devices/ncs:device/ncs:name";
                }
                must "count(../../../../../../admin-domain/l2-gateway-groups/l2-gateway-
group/devices/device[name = current()/../name]) = 1" {
                  error-message "A device can only belong to one l2 gateway group.";
                  tailf:dependency ".";
                }
                must "/vts:cisco-vts/vts:global-settings/vts:anycast-gateway/vts:anycast-
gw-address" {
                  error-message "L2 gateway requires an anycast gateway mac address.";
                  tailf:dependency "/vts:cisco-vts/vts:global-settings/vts:anycast-
gateway/vts:anycast-gw-address";
                }
              } // leaf name
              leaf gwgroup-change-timestamp {
                type ietf-yang:date-and-time;
                description "The timestamp when the gateway group change was
triggered";
                tailf:info  "The timestamp when the gateway group change was
triggered";
              } // leaf gwgroup-change-timestamp
            } // list device
          } // container devices

          choice parent {
            case ad-l3-gw-parent-case {
```

```
            leaf ad-l3-gw-parent {
              description "This group's L3 gateway parent in " +
                          "the current administrative domain " +
                          "in the infrastructure policy hierarchy.";
              tailf:info "L3 gateway in current admin domain.";
              type leafref {
                path "../../../l3-gateway-groups/l3-gateway-group/name";
              }
            }
          } // l3-gw-parent-case
          case l3-gw-parent-case {
            leaf l3-gw-parent {
              description "This group's L3 gateway parent in the infrastructure policy
hierarchy.";
              tailf:info "L3 gateway.";
              type leafref {
                path "../../../../../l3-gateway-groups/l3-gateway-group/name";
              }
            }
          } // l3-gw-parent-case
        } // choice parent
        container dc-gw-parameters {
          presence "Creates dc gw parameters.";
          description "Parameters for the DC gw reference.";
          tailf:info "Parameters for the DC gw reference.";
          uses ncs:service-data;
          ncs:servicepoint l3-dc-gw-parameters-servicepoint;
          choice dc-gw-parent {
            case l3-dc-gw-parent-case {
              leaf l3-dc-gw-parent {
                description "This group's L3 datacenter gateway parent in the
infrastructure policy hierarchy.";
                tailf:info "L3 datacenter gateway.";
                type leafref {
                  path "../../../../../../l3-dc-gateway-groups/l3-dc-gateway-group/name";
                }
              }
            } // case l3-dc-gw-parent-case
          } // choice dc-gw-parent
        }// container dc-gw-parameters
        container policy-parameters {
          description "Nodes in this container control the " +
                      "behavior of an L2 group of " +
                      "the infrastructure policy.";
          tailf:info "Policy parameters.";
          leaf distribution-mode {
            description
              "Mode of distribution.";
            tailf:info "Distribution mode.";
            type identityref {
```

```
            base "vts-ids:l2-distribution-mode";
           }
          }
          uses GROUP-POLICY-PARAMS {
           refine "packet-replication" {
            must "count(../../../l2-gateway-group/policy-parameters/packet-
replication != current()/../packet-replication) = 0" {
             error-message "L2 gateway groups within the same administrative " +
                     "domain must use the same packet replication mechanism.";
             tailf:dependency ".";
            }
           }
          } // uses GROUP-POLICY-PARAMS
         } // container policy-parameters
        } // list l2-gateway-groups
       } // container l2-gateway-groups

       container l3-gateway-groups {
        description "Set of layer 3 gateway groups.";
        tailf:info "Layer 3 gateway groups.";
        list l3-gateway-group {
         description "List of layer 3 gateway groups.";
         tailf:info "Layer 3 gateway group.";
         key name;
         leaf name {
          description "Name of the L3 gateway group.";
          tailf:info "Name.";
          type vts-types:string128;
         }
         leaf current-dc-gw-parent {
          description "Name of the current L3 DC gateway group parent chosen to
achieve load balancing.";
          tailf:info "Name of current L3 DC gateway group parent.";
          type vts-types:string128;
         }
         leaf description {
          description "Description of the layer 3 gateway group.";
          tailf:info "Description.";
          type vts-types:string128;
         }
         choice parent {
          case l3-dc-gw-parent-case {
           container l3-dc-gw-parents {
            description "This groups's set of (L3 datacenter gateway group) parents in
the infrastructure policy hierarchy. " +
                    "When multiple parents are defined, they are used for load
balancing. ";
            tailf:info "L3 datacenter gateway group parents.";

            list l3-dc-gw-parent {
```

```
                description "List of this groups's set of L3 datacenter gateway parents in
the infrastructure policy hierarchy.";
                tailf:info "L3 datacenter gateway group parent.";

                key name;
                leaf name {
                  description "A L3 datacenter gateway group acting as parent of this
group in the infrastructure policy hierarchy.";
                  tailf:info "Name.";
                  type leafref {
                    path "../../../../../../l3-dc-gateway-groups/l3-dc-gateway-
group/name";
                  }
                }
              } // list l3-dc-gw-parent
            } // container l3-dc-gw-parents
          } // case l3-dc-gw-parent-case
        } // choice parent

        container devices {
          description "Set of layer 3 devices, acting as site-wide data center
gateways.";
          tailf:info "Devices.";
          list device {
            uses ncs:service-data;
            ncs:servicepoint l3-gateway-group-servicepoint;
            key name;
            leaf name {
              description "Set of layer 3 devices, acting as layer 3 gateways.";
              tailf:info "Name.";
              type leafref {
                path "/ncs:devices/ncs:device/ncs:name";
              }
              must "count(../../../../../../admin-domain/l3-gateway-groups/l3-gateway-
group/devices/device[name = current()/../name]) = 1" {
                error-message "A device can only belong to one l3 gateway group.";
                tailf:dependency ".";
              }
            } // leaf name
            leaf gwgroup-change-timestamp {
              type ietf-yang:date-and-time;
              description "The timestamp when the gateway group change was
triggered";
              tailf:info  "The timestamp when the gateway group change was
triggered";
            } // leaf gwgroup-change-timestamp
          } // list device
        } // container devices
        uses GROUP-POLICY-PARAMS-L3GW;
      } // list l3-gateway-groups
```

```
        } // container l3-gateway-groups
      } // list admin-domain
    } // container admin-domains

    container l3-gateway-groups {
      description "Set of layer 3 gateway groups.";
      tailf:info "Layer 3 gateway groups.";
      list l3-gateway-group {
        description "List of layer 3 gateway groups.";
        tailf:info "Layer 3 gateway group.";
        key name;
        leaf name {
          description "Name of the L3 gateway group.";
          tailf:info "Name.";
          type vts-types:string128;
        }
        leaf current-dc-gw-parent {
        description "Name of the current L3 DC gateway group parent chosen to
achieve load balancing.";
          tailf:info "Name of current L3 DC gateway group parent.";
          type vts-types:string128;
        }
        leaf description {
          description "Description of the layer 3 gateway group.";
          tailf:info "Description.";
          type vts-types:string128;
        }
        choice parent {
          case l3-dc-gw-parent-case {
            container l3-dc-gw-parents {
              description "This groups's set of (L3 datacenter gateway group) parents in
the infrastructure policy hierarchy. " +
                          "When multiple parents are defined, they are used for load
balancing. ";
              tailf:info "L3 datacenter gateway group parents.";

              list l3-dc-gw-parent {
                description "List of this groups's set of L3 datacenter gateway parents in
the infrastructure policy hierarchy.";
                tailf:info "L3 datacenter gateway group parent.";

                key name;
                leaf name {
                  description "A L3 datacenter gateway group acting as parent of this
group in the infrastructure policy hierarchy.";
                  tailf:info "Name.";
                  type leafref {
                    path "../../../../../l3-dc-gateway-groups/l3-dc-gateway-group/name";
                  }
                }
```

```
            } // list l3-dc-gw-parent
          } // container l3-dc-gw-parents
        } // case l3-dc-gw-parent-case
      } // choice parent

      container devices {
        description "Set of layer 3 devices, acting as layer 3 gateways.";
        tailf:info "Devices.";
        list device {
          uses ncs:service-data;
          ncs:servicepoint l3-gateway-group-servicepoint;
          key name;
          leaf name {
            description "Name of the device.";
            tailf:info "Name.";
            type leafref {
              path "/ncs:devices/ncs:device/ncs:name";
            }
            must "count(../../../../l3-gateway-group/devices/device[name =
current()/../name]) = 1" {
              error-message "A device can only belong to one l3 gateway group.";
              tailf:dependency ".";
            }
          } // leaf name
          leaf gwgroup-change-timestamp {
            type ietf-yang:date-and-time;
            description "The timestamp when the gateway group change was triggered";
            tailf:info  "The timestamp when the gateway group change was triggered";
          } // leaf gwgroup-change-timestamp
        } // list device
      } // container device
      uses GROUP-POLICY-PARAMS-L3GW;
    } // list l3-gateway-groups
  } // container l3-gateway-groups

  container l3-dc-gateway-groups {
    description "Set of layer 3 gateway groups, acting " +
                "as data center gateways at the site level.";
    tailf:info "Layer 3 gateway (site DCGW) groups.";
    list l3-dc-gateway-group {
      description "List of layer 3 gateway groups, acting " +
                  "as data center gateways. " +
                  "Devices in a group must be configured to provide redundancy among
themselves";
      tailf:info "Layer 3 gateway (site DCGW) group.";
      key name;
      leaf name {
        description "Name of the L3 gateway group " +
                    "(data center gateway).";
        tailf:info "Name.";
```

```
        type vts-types:string128;
      }
      leaf description {
        description "Description of the layer 3 data center gateway group.";
        tailf:info "Description.";
        type vts-types:string128;
      }
      leaf l3-dci-parent {
        description "This group's datacenter gateway interconnect parent in the
infrastructure policy hierarchy.";
        tailf:info "DCI.";
        type leafref {
          path "../../../l3-dci-groups/l3-dci-group/name";
        }
        mandatory true;
      }

      container devices {
        description  "Set of layer 3 devices, acting as site-wide data center gateways.";
        tailf:info "Devices.";
        list device {
          uses ncs:service-data;
          ncs:servicepoint dcgw-gateway-group-servicepoint;
          description "List of the devices.";
          tailf:info "device.";

          key name;
          leaf name {
            description "Name of the device.";
            tailf:info "Name.";
            type leafref {
              path "/ncs:devices/ncs:device/ncs:name";
            }

            must "count(../../../../l3-dc-gateway-group/devices/device[name =
current()/../name]) = 1" {
              error-message "A device can only belong to one l3 site-wide data center
gateway group.";
              tailf:dependency ".";
            }
          } // leaf name
          leaf gwgroup-change-timestamp {
            type ietf-yang:date-and-time;
            description "The timestamp when the gateway group change was triggered";
            tailf:info  "The timestamp when the gateway group change was triggered";
          } // leaf gwgroup-change-timestamp
        }// list device
      }//devices

      container policy-parameters {
```

```
      description "Nodes in this container control the " +
                  "behavior of an L2 group of " +
                  "the infrastructure policy.";
      tailf:info "Policy parameters.";
      leaf control-plane-protocol {
        description
          "Control plane protocol used to learn end point addresses.";
        tailf:info "Control plane protocol.";
        type identityref {
          base "vts-ids:control-plane-protocol";
        }
      } // leaf control-plane-protocol
    } // container policy-parameters
  } // list l3-gateway-site-dcgw-group
} // container l3-gateway-site-dcgw-groups



container l3-dci-groups {
  description "Set of layer 3 gateway groups, acting " +
              "as data center interconnects.";
  tailf:info "Layer 3 gateway (DCI) groups.";
  list l3-dci-group {
    description "List of layer 3 gateway groups, acting " +
                "as data center interconnects.";
    tailf:info "Layer 3 gateway (DCI) group.";
    key name;
    leaf name {
      description "Name of the L3 gateway group " +
                  "(data center gateway).";
      tailf:info "Name.";
      type vts-types:string128;
    }
    leaf description {
      description "Description of the layer 3 data center interconnect group.";
      tailf:info "Description.";
      type vts-types:string128;
    }

    container devices {
      description "Set of layer 3 devices, acting as data center interconnects.";
      tailf:info "Devices.";
      list device {
        uses ncs:service-data;
        ncs:servicepoint dci-gateway-group-servicepoint;
        key name;
        leaf name {
          description "Name of the device.";
          tailf:info "Name.";
          type leafref {
```

```
              path "/ncs:devices/ncs:device/ncs:name";
             }
             must "count(../../../../l3-dci-group/devices/device[name = current()/../name])
= 1" {
              error-message "A device can only belong to one data center interconnect
group.";
              tailf:dependency ".";
             }
            } // leaf name
            leaf gwgroup-change-timestamp {
             type ietf-yang:date-and-time;
             description "The timestamp when the gateway group change was triggered";
             tailf:info  "The timestamp when the gateway group change was triggered";
            } // leaf gwgroup-change-timestamp
           } // list device
         } // container devices

         uses  DCI-GROUP-REDUNDANCY-SETTINGS;

         container stitching-profiles {
          description "Set of stitching profiles.";
          tailf:info "Stitching Profiles.";
          list stitching-profile {
           uses ncs:service-data;
           ncs:servicepoint dci-gateway-group-stitching-servicepoint;
           key id;
           leaf id {
            description "Id of the stitching profile.";
            tailf:info "Id.";
            type leafref {
             path "/vts-profiles:profiles/vts-profiles:profile/vts-profiles:id";
            }
            //must "/vts-profiles:profiles/vts-profiles:profile[id = current()]/type = 'vts-
profiles:stitching'";
           }
           uses vts-types:MODIFIED-TIME-ELEMENT;
          } // list stitching-profile
         }// stitching-profiles
        } // list l3-gateway-dci-group
       } // container l3-gateway-dci-groups
     } // container infra-policy
   }  // grouping INFRA-POLICY
}
```

6.7 Device Extension

6.7.1 N9K

```
module n9k-extension {
 namespace "http://cisco.com/ns/yang/services/n9k-extension";
 prefix n9k-extension;

 import ietf-inet-types {
  prefix inet;
 }
 import ietf-yang-types {
  prefix yang;
 }
 import tailf-common {
  prefix tailf;
 }
 import tailf-ncs {
  prefix ncs;
 }
 import cisco-vts {
  prefix vts;
 }
 import cisco-vts-types {
  prefix vts-types;
 }
 import cisco-vts-identities {
  prefix cisco-vts-identities;
 }

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the management
   of the Cisco N9K devices.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

 revision "2015-02-28" {
        description
    "Initial revision.";
 }
```

```
 augment "/ncs:devices/ncs:device" {
         when "ncs:device-type/ncs:cli/ncs:ned-id='cisco-nx-id:cisco-nx'";

 container device-info {

  presence "Creates device info";

  // TODO: remove. This is a copy of another leaf in the same container w/ the same
name

  uses vts:VTS-DEVICE-METADATA;
  leaf name {
   tailf:info "Device name ( Switch )";
   type leafref  {
    path "/ncs:devices/ncs:device/ncs:name";
   }
  }

   leaf platform {
    // TODO: reconsider having a leaf saying this is a N9K on a N9k-specific
extension
    //       consider we may different subtypes of N9K platforms
    description "Device platform.";
    tailf:info "Platform.";
          type identityref {
     base "cisco-vts-identities:switch-platform";
    }
  }
  leaf OS {
    description "Operating System.";
    tailf:info "Operating System.";
          type vts-types:string128;
  }
  leaf version {
    description "Operating System version.";
    tailf:info "Operating System version.";
    type string;
  }
  leaf device-use  {
    description "Role of the switch. How it is being used.";
    tailf:info "Usage (role) of the switch.";
    type identityref {
     base "cisco-vts-identities:switch-role";
    }
  }
  leaf group-id {
     tailf:info "Group id";
     type string;
  }
```

```
leaf peering-mode {
    tailf:info "Mode in which DC-GW operates";
    type enumeration { enum VRF-PEERING; enum INTEGRATED; }
}
leaf vrf-capacity {
    tailf:info "Vrf Capacity of the device";
    default 900;
    type uint32;
}
container bgp-peering-info {
  description "BGP peering information.";
  tailf:info "BGP peering information.";
  leaf bgp-asn {
    description "BGP ASN number.";
    tailf:info "BGP ASN number.";
    type uint32;
  }
  leaf loopback-if-num {
    description "BGP loopback interface number.";
    tailf:info "BGP loopback interface number.";
    type uint16;
  }
  leaf loopback-if-ip {
    description "BGP loopback interface IP address.";
    tailf:info "A.B.C.D/LEN;;IP prefix and network mask " +
               "length in format x.x.x.x/m";
    type inet:ip-prefix;
  }
}
list port {
  description "List of switch ports.";
  tailf:info "Switch port.";
  key name;

  uses vts:COMMON-PHYSICAL-PORT;

        leaf port-channel-membership {
    // TODO: remove
    description "Port channel membership.";
    tailf:info "Port channel membership.";
    type vts-types:string128;
          // TODO: consider semantics of this default
          default "none";
  }
  leaf remote-interface-name {
    description "Interface name of the remote node connected " +
                "to the port.";
    tailf:info "Interface name of the remote node connected " +
                "to the port.";
    type vts-types:string128;
```

```
        }

  leaf remote-mac {
    description "MAC address of the remote node connected " +
                " to the port.";
    tailf:info "Remote MAC address.";
          type yang:mac-address;
  }
        leaf remote-type {
    description "Type of the remote server connected " +
                "to the port.";
    tailf:info "Type of the remote server connected " +
                "to the port.";
    type identityref {
      base "cisco-vts-identities:server-type";
    }
        }
  list remote-server-id {
    key name;
    leaf name {
      description "Identifier of the remote server connected " +
                  "to the port.";
      tailf:info "ID of the remote server connected " +
                  "to the port.";
      type vts-types:string128;
    }
  }
} //port
container vpc {
  leaf vpc-id {
    tailf:info ";;<-1000-1000>  Domain id.";
    type int16 {
      range "-1000..1000";
    }
  }
  container vpc-peer {
    leaf vpc-peer-ip {
      tailf:info "Specify destination ip address of peer switch.";
      type inet:ipv4-address;
    }
    leaf vpc-peer-name {
      tailf:info "Specify peer switch name.";
      type vts-types:string128;
    }
  }
  leaf vpc-peer-link {
    description "Peer link of the virtual port channel.";
    tailf:info "<1-4096>;;Port Channel number.";
    type uint16 {
      range "1..4096";
```

```
    }
  }
} // container vpc
list port-channel {
  description "List of port channel the switch belongs to.";
  tailf:info "Port channel.";
        key name;
  leaf name {
    description "Port channel name.";
    tailf:info "<1-4096>;;Port Channel number.";
    // TODO: reconsider this into an integer
    type uint16 {
      range "1..4096";
    }
  }
        leaf-list ports {
    description "Set of ports belonging to the port channel.";
    tailf:info "Port inthe port channel.";
          type vts-types:string128;
        }
        leaf type {
    description "Type of port channel.";
    tailf:info "Type.";
    type identityref {
      base "cisco-vts-identities:port-channel-type";
    }
        }
} //port-channel-list

container servers {
  description "Servers connected to the switch.";
  tailf:info "Servers.";

        list server {
    description "List of servers connected to the switch.";
    tailf:info "Server.";
                key server-id;
                leaf server-id {
      description "Server identifier.";
      tailf:info "ID.";
                  type vts-types:string128;
                }
    list nic {
      description "List of network controller interfaces on " +
                  "the server.";
      tailf:info "NIC.";
                  key id;
                  leaf id {
        description "Identifier of the network controller " +
                    "interface";
```

```
        tailf:info "ID.";
                        type vts-types:string128;
                }
                leaf mac {
        // TODO: note how the same data is in multiple places
        description "MAC address of the network controller " +
                "interface";
        tailf:info "MAC.";
        type leafref {
          path "../../../../port/remote-mac" ;
        }
                }
                leaf ip {
        description "IP address of the network controller " +
                "interface";
        tailf:info "IP.";
                        type inet:ip-address;
                }
      leaf remote-mac {
                  // TODO: note how the same data is in multiple places
        description "MAC address of the switch port the " +
                "server NIC is connected to";
        tailf:info "Switch port MAC.";
        type leafref {
          path "../../../../port/local-mac" ;
        }
                }
               } //nic
    } //server
   } //servers
   leaf device-group {
     description "Refers the device-group";
     tailf:info "Refers the device-group";
     type leafref {
       path "/vts:cisco-vts/vts:device-groups/vts:device-group/vts:name";
     }
   } //device-group
  } //device-info
  } //augment
} //module
```

6.7.2 ASR9K

```
module asr9k-extension {
  namespace "http://cisco.com/ns/yang/services/asr9k-extension";
  prefix asr9k-extension;

  import ietf-inet-types {
```

```
  prefix inet;
}
import ietf-yang-types {
  prefix yang;
}
import tailf-common {
  prefix tailf;
}
import tailf-ncs {
  prefix ncs;
}
import cisco-vts {
  prefix vts;
}
import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix cisco-vts-identities;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the management
   of the Cisco ASR9K devices.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-02-28" {
        description
    "Initial revision.";
}

augment "/ncs:devices/ncs:device" {
        when "ncs:device-type/ncs:cli/ncs:ned-id='cisco-ios-xr-id:cisco-ios-xr'";

container device-info {
```

```
    presence "Creates device info";

    // TODO: remove. This is a copy of another leaf in the same container w/ the same
name
    uses vts:VTS-DEVICE-METADATA;
    leaf name {
      tailf:info "Device name ( Switch )";
      type leafref  {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }
    leaf platform {
      // TODO: reconsider having a leaf saying this is a ASR9K on a ASR9k-specific
extension
      //       consider we may different subtypes of ASR9K platforms
      description "Device platform.";
      tailf:info "Platform.";
            type identityref {
        base "cisco-vts-identities:switch-platform";
      }
            default cisco-vts-identities:ASR9K;
    }
    leaf pending-sync {
      description "Does the device need to be synced?";
      tailf:info "Sync status.";
      type empty;
    }
    leaf OS {
      description "Operating System.";
      tailf:info "Operating System.";
            type vts-types:string128;
    }
    leaf version {
      description "Operating System version.";
      tailf:info "Operating System version.";
      type string;
    }
    leaf device-use  {
      description "Role of the switch. How it is being used.";
      tailf:info "Usage (role) of the switch.";
      type identityref {
        base "cisco-vts-identities:switch-role";
      }
    }
    leaf group-id {
       tailf:info "Group id";
       type string;
    }
    leaf peering-mode {
       tailf:info "Mode in which DC-GW operates";
```

```
    type enumeration { enum VRF-PEERING; enum INTEGRATED; }
  }
  leaf vrf-capacity {
    tailf:info "Vrf Capacity of the device";
    default 900;
    type uint32;
  }
  container bgp-peering-info {
   description "BGP peering information.";
   tailf:info "BGP peering information.";
   leaf bgp-asn {
    description "BGP ASN number.";
    tailf:info "BGP ASN number.";
    type uint32;
   }
   leaf loopback-if-num {
    description "BGP loopback interface number.";
    tailf:info "BGP loopback interface number.";
    type uint16;
   }
   leaf loopback-if-ip {
    description "BGP loopback interface IP address.";
    tailf:info "A.B.C.D/LEN;;IP prefix and network mask " +
               "length in format x.x.x.x/m";
    type inet:ip-prefix;
   }
  }
  list port {
   description "List of switch ports.";
   tailf:info "Switch port.";
   key name;

   uses vts:COMMON-PHYSICAL-PORT;

         leaf port-channel-membership {
    // TODO: remove
    description "Port channel membership.";
    tailf:info "Port channel membership.";
    type vts-types:string128;
          // TODO: consider semantics of this default
          default "none";
   }
   leaf remote-interface-name {
    description "Interface name of the remote node connected " +
                "to the port.";
    tailf:info "Interface name of the remote node connected " +
                "to the port.";
    type vts-types:string128;
         }
```

```
  leaf remote-mac {
    description "MAC address of the remote node connected " +
                " to the port.";
    tailf:info "Remote MAC address.";
          type yang:mac-address;
  }
        leaf remote-type {
    description "Type of the remote server connected " +
                "to the port.";
    tailf:info "Type of the remote server connected " +
                "to the port.";
    type identityref {
      base "cisco-vts-identities:server-type";
    }
        }
  list remote-server-id {
    key name;
    leaf name {
      description "Identifier of the remote server connected " +
                  "to the port.";
      tailf:info "ID of the remote server connected " +
                  "to the port.";
      type vts-types:string128;
    }
  }
} //port
container vpc {
  leaf vpc-id {
    tailf:info ";;<-1000-1000>  Domain id.";
    type int16 {
      range "-1000..1000";
    }
  }
  container vpc-peer {
    leaf vpc-peer-ip {
      tailf:info "Specify destination ip address of peer switch.";
      type inet:ipv4-address;
    }
    leaf vpc-peer-name {
      tailf:info "Specify peer switch name.";
      type vts-types:string128;
    }
  }
  leaf vpc-peer-link {
    description "Peer link of the virtual port channel.";
    tailf:info "<1-4096>;;Port Channel number.";
    type uint16 {
      range "1..4096";
    }
  }
```

```
} // container vpc
list port-channel {
  description "List of port channel the switch belongs to.";
  tailf:info "Port channel.";
        key name;
  leaf name {
    description "Port channel name.";
    tailf:info "<1-4096>;;Port Channel number.";
    // TODO: reconsider this into an integer
    type uint16 {
      range "1..4096";
    }
  }
        leaf-list ports {
    description "Set of ports belonging to the port channel.";
    tailf:info "Port inthe port channel.";
          type vts-types:string128;
        }
        leaf type {
    description "Type of port channel.";
    tailf:info "Type.";
    type identityref {
      base "cisco-vts-identities:port-channel-type";
    }
        }
} //port-channel-list

container servers {
  description "Servers connected to the switch.";
  tailf:info "Servers.";

        list server {
    description "List of servers connected to the switch.";
    tailf:info "Server.";
                key server-id;
                leaf server-id {
      description "Server identifier.";
      tailf:info "ID.";
                  type vts-types:string128;
                }
    list nic {
      description "List of network controller interfaces on " +
                  "the server.";
      tailf:info "NIC.";
                  key id;
                  leaf id {
        description "Identifier of the network controller " +
                    "interface";
        tailf:info "ID.";
                        type vts-types:string128;
```

```
                }
                leaf mac {
        // TODO: note how the same data is in multiple places
        description "MAC address of the network controller " +
                  "interface";
        tailf:info "MAC.";
        type leafref {
          path "../../../../port/remote-mac" ;
        }
                }
                leaf ip {
        description "IP address of the network controller " +
                  "interface";
        tailf:info "IP.";
                        type inet:ip-address;
                }
       leaf remote-mac {
                  // TODO: note how the same data is in multiple places
        description "MAC address of the switch port the " +
                  "server NIC is connected to";
        tailf:info "Switch port MAC.";
        type leafref {
          path "../../../../port/local-mac" ;
        }
                }
               } //nic
    } //server
   } //servers

   leaf device-group {
     description "Refers the device-group";
     tailf:info "Refers the device-group";
     type leafref {
       path "/vts:cisco-vts/vts:device-groups/vts:device-group/vts:name";
     }
   } //device-group
  } //device-info
  } //augment
} //module
```

6.7.3 Device Extension Infra

```
module device-extension-infra-policy {
  namespace "http://cisco.com/ns/yang/services/device-extension-infra-policy";
  prefix dev-infra-policy;

  import tailf-common {
    prefix tailf;
  }
```

```
 import tailf-ncs {
   prefix ncs;
 }

 import cisco-vts {
   prefix vts;
 }

 organization "Cisco Systems, Inc.";

 contact
   "Cisco Systems, Inc.
    Customer Service

    Postal: 170 West Tasman Drive
    San Jose, CA 95134

    Tel: +1 800 533-NETS";

 description
   "This module contains a collection of YANG definitions
    for Cisco's VTS's management. Specifically, for the management
    of the NCS devices in conjunction with the infrastructure policy.

    Copyright (c) 2015 by Cisco Systems, Inc.
    All rights reserved.";

 revision "2015-06-21" {
   description
     "Initial revision.";
 }

 augment "/ncs:devices/ncs:device" {
   leaf l2-gateway-group {
     description "Layer 2 gateway group this device belongs to.";
     tailf:info "L2 gateway group.";
     type leafref {
       path "/vts:cisco-vts/vts:infra-policy/vts:admin-domains/vts:admin-domain/vts:l2-
gateway-groups/vts:l2-gateway-group/vts:name";
     }
   } // leaf l2-gateway-group
 } //augment
} //module
```

6.8 Inventory

```
submodule cisco-vts-inventory {

 belongs-to cisco-vts {
```

```
  prefix vts;
}

import ietf-inet-types {
  prefix inet;
}

import tailf-common {
  prefix tailf;
}
import tailf-ncs {
  prefix ncs;
}
import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix vts-ids;
}
import ietf-yang-types {
  prefix ietf-yang;
}
include cisco-vts-common {
  revision-date 2017-04-20;
}
include cisco-vts-device-operational-data {
  revision-date 2015-06-15;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the inventory
   of the physical topology.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2016-08-30" {
  description
```

```
    "VTS 2.3.1
     Enhancements:
        - Added leaf underlay-vlan-id to vtfs container, which tracks vlan-id used by
the underlay bridge/port group
    - Added leaf disable-validations to disable ports and admin domain validations
     Corrections:
    - Moved connid leaf from fex ports to fex

   ";
 }

 revision "2016-07-15" {
  description
   "VTS 2.3.0
    Enhancements:
     - Added leaves name, mode, interfacem and container
         bonds under vtf list
     - Added leaves for timestamping the creation of entities:
         vtf-creation-timestamp and
         devicegroup-device-creation-timestamp
    Corrections:
     - Added unique statements for local-mac and
         binding-host-name under vtf list
   ";

 }

 revision "2015-06-11" {
  description
    "Initial revision.";
 }


 grouping VTS-INVENTORY {
  container devices {
    description "Devices that VTS manages as part of the inventory.";
    tailf:info "VTS device inventory.";
    list device {
      description "List of switches with the servers and other " +
                  "devices connected to them.";
      tailf:info "Device (Switch) Port mapping.";
      ncs:servicepoint portserver;
      uses ncs:service-data;
      key name;
      leaf name {
        description "Name of switch that is part of VTS inventory.";
        tailf:info "Device (Switch) name.";
        type leafref  {
          path "/ncs:devices/ncs:device/ncs:name";
        }
```

```
}
leaf inventory-trigger-timestamp {
  type ietf-yang:date-and-time;
  description "The timestamp when inventory is triggered";
  tailf:info  "The timestamp when inventory is triggered";
}
uses VTS-DEVICE-NETWORK-OPER-DATA;
container ports {
  description "Device interfaces that VTS manages as part of " +
              "inventory.";
  tailf:info "VTS port inventory.";
  list port {
    description "List of physical portson the switch that VTS " +
                "manages as part of inventory.";
    tailf:info "Device (switch) port.";
    key name;
    uses vts:COMMON-PHYSICAL-PORT;
    choice connnected-entities {
      description "A port/port-channel can have connected to it " +
                  "either servers of fabric extenders.";
      case connected-servers {
        description "Servers connected to a port/port-channel.";
        container servers {
          description "Servers that VTS manages as part of inventory.";
          tailf:info "VTS server inventory.";
          list server {
            description "List of servers attached to device interfaces " +
                        "that is managed by VTS.";
            tailf:info "VTS server inventory.";
            key name;
            uses vts:COMMON-SERVER;
            leaf connid {
              tailf:info "Unique ID for port server mapping";
              type vts-types:uuid;
            } // connid

            leaf vtf_ip {
              tailf:info "VTF VM IP address";
              type inet:ip-address;
            }
          } // server
        } // servers
      } // case connected-servers
      case connected-fexs {
        description "Fabrix extenders connected to a port/port-channel.";
        container fexs {
          description "Set of FEXs under a physical port of port channel.";
          tailf:info "Fabric extenders.";
          list fex {
            description "List of FEXs under a physical port or port channel.";
```

```
                tailf:info "Fabric extender.";
                key fex-id;
                leaf fex-id {
                  description "Fex identifier.";
                  tailf:info "Id.";
                  type vts-types:fex-identifier;
                }
                leaf connid {
                  description "ID for the port-fex mapping.";
                  tailf:info "Unique ID for port fex mapping";
                  type vts-types:uuid;
                } //connid
                container ports {
                  description "Set of host interfaces on the FEX.";
                  tailf:info "Ports.";
                  list port {
                    description "List of host interfaces on the FEX.";
                    tailf:info "Port.";
                    key name;
                    uses vts:COMMON-PHYSICAL-PORT;
                    container servers {
                      description "Set of servers connected to this host interface on the
FEX.";
                      tailf:info "Servers.";
                      list server {
                        description "List of servers connected to this host interface on the
FEX.";
                        tailf:info "Server.";
                        key name;
                        uses vts:COMMON-SERVER;
                        leaf connid {
                          description "ID for the port-server mapping.";
                          tailf:info "Unique ID for port server mapping";
                          type vts-types:uuid;
                        } // connid
                      } // server
                    } // servers
                  } // port
                } // ports
              } // fex
            } // fexs
          } // case connected-fexs
        } // choice connnected-entities
      } // port
    } // ports
  } // device
} // devices

container vtfs {
  list vtf {
```

```
description "List of VTF VM ips under this server.
             If this list is non-empty, it indicates that
             VTF VMs have been spawned on this server";
tailf:info "List of VTF VM ids under this server.";
ncs:servicepoint vtf-registration-servicepoint;
uses ncs:service-data;

key ip;
unique "local-mac";
unique "binding-host-name";

leaf ip {
  tailf:info "VTF VM IP address";
  type inet:ip-address;
}

leaf name {
  description "VTF VM Name.";
  tailf:info "VTF VM Name.";
  type vts-types:string128;
}

leaf local-mac {
  tailf:info "Local MAC address";
  //type yang:mac-address;
  type string {
    pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
  }
}
leaf username {
  tailf:info "Username of VTF instance";
  mandatory true;
  type string;
}
leaf password {
  tailf:info "Encrypted password of VTF instance";
  type string;
}
leaf gateway-ip {
  description "Gateway/Next Hop IP for the VTF VM";
  tailf:info "Gateway/Next Hop IP for the VTF VM";
  type inet:ip-address;
}
leaf binding-host-name {
  description "Compute Host identifier";
  tailf:info "Compute Host identifier";
  mandatory true;
  type string;
  //type leafref {
  //  path "../../../devices/device/ports/port/servers/server/name";
```

```
        //}
      }
      leaf vpp-client-name {
        description "Optional client name for VPP";
        tailf:info "Optional client name for VPP";
        type vts-types:string128;
      }

      leaf device-group {
        description "Refers the device-group";
        tailf:info "Refers the device-group";
        type leafref {
          path "../../../device-groups/device-group/name";
        }
      } //device-group
      leaf subnet-mask {
        description "Subnet mask.";
        tailf:info "Subnet mask.";
        type inet:ip-address;
      }
      leaf underlay-network {
        description "Name of underlay network portgroup/bridge on " +
                    "binding-host which VTF is attached to.";
        tailf:info  "Name of underlay network portgroup/bridge on " +
                    "binding-host which VTF is attached to.";
        type string;
      }
      leaf tenant-network {
        description "Name of tenant network portgroup/bridge on " +
                    "binding-host which VTF is attached to.";
        tailf:info  "Name of tenant network portgroup/bridge on " +
                    "binding-host which VTF is attached to.";
        type string;
      }
      leaf underlay-vlan-id {
        description "Vlan id that will be tagged on underlay network portgroup/bridge
on " +
                    "binding-host which VTF is attached to.";
        tailf:info  "Vlan id that will be tagged on underlay network portgroup/bridge.";
        type vts-types:vlan-id;
      }
      leaf datastore {
        description "Name of datastore on binding-host.";
        tailf:info "Name of datastore on binding-host.";
        type string;
      }
      leaf mode {
        description "Deployment mode of VTF.";
        tailf:info "Deployment mode of VTF.";
        type identityref {
```

```
        base "vts-ids:vtf-mode";
       }
       default vts-ids:VM;
      }
      container vtf-vhost-interfaces {
       description "Set of VTF vhost interfaces.";
       tailf:info "Set of VTF vhost interfaces.";

       list vtf-vhost-interface {
        description "List of VTF vhost interfaces.";
        tailf:info "List of VTF vhost interfaces.";
        key name;
        max-elements 1;

        leaf name {
         description "Name of physical or bond interface on the compute host.";
         tailf:info "Name of physical or bond interface on the compute host.";
         type string;
         must "(../../../mode='vts-ids:vhost')" {
          error-message "VTF vhost interface name is only applicable for vtf in vhost
mode.";
          tailf:dependency "../../../mode";
         }
        }

        leaf pci-driver {
         description "Pci driver to use for this interface.";
         tailf:info "Pci driver to use for this interface.";
         type identityref {
          base "vts-ids:pci-driver-type";
         }
         default vts-ids:uio_pci_generic;
        }

        uses BONDED-INTERFACE {
         refine "bond" {
          must "(../../../mode='vts-ids:vhost')" {
           error-message "VTF vhost bond interface is only applicable for vtf in
vhost mode.";
           tailf:dependency "../../../mode";
          }
         }

         refine "bond/bond-interfaces/bond-interface/name" {
          must "current() != ../../../../name" {
           error-message "This interface cannot be added to the bond since it is
already added as VTF vhost interface.";
           tailf:dependency "../../../../name";
          }
         }
```

```
        } // uses BONDED-INTERFACE
       } // list vtf-vhost-interface
      } // container vtf-vhost-interfaces
      container reachable-subnets {
       description "Set of subnets reachable through gateway.";
       tailf:info "Set of subnets reachable through gateway.";
       list reachable-subnet {
        description "List of subnets reachable through gateway.";
        tailf:info "List of subnets reachable through gateway.";
        key subnet;

        leaf subnet {
         description "Subnetwork address.";
         tailf:info "Subnetwork address.";
         type tailf:ip-address-and-prefix-length;
        }
       } // list reachable-subnet
      } // container reachable-subnets
     } // list vtf
    } // container vtf

    container xrvr-groups {
     list xrvr-group {
      description "List of XRVR device groups, consisting of XRVR's  and the
VTF-VMs attached to them ";
      tailf:info "XRVR to VTF Mapping.";
      ncs:servicepoint xrvr-to-vtf-map-servicepoint;
      uses ncs:service-data;

      key group-name;
      leaf group-name {
       description "Name of xrvr group that VTF's will belong to. Each VTF
will belong to all XRVR's in the group.";
       tailf:info "XRVR group name.";
       type vts-types:string128;
      }

      container xrvr-devices {
       list xrvr-device {
        description "List of XRVR devices";
        tailf:info "XRVR in Group";
        key name;
        leaf name {
         description "Name of switch that is part of VTS inventory.";
         tailf:info "Device (Switch) name.";
         type leafref  {
          path "/ncs:devices/ncs:device/ncs:name";
         }
        }
       } // list xrvr-device
```

```
      } //container xrvr-devices

      container vtfs {
        list vtf {
          description "List of VTF VMs managed by the XRVR group";
          tailf:info "VTF VMs that are managed by the XRVR group";
          key ip;

          leaf ip {
            description "IP address of the VTF instance";
            tailf:info "IP address of the VTF instance";
            type inet:ip-address;
          }

          leaf vtf-creation-timestamp {
            type ietf-yang:date-and-time;
            description "The timestamp when the vtf was triggered";
            tailf:info  "The timestamp when the vtf was triggered";
          }
        } // list vtf
      } // container vtfs
    } // list xrvr-group
  } // container xrvr-devices

} // grouping VTS-INVENTORY

grouping VTS-NETWORK-TO-PORTS-MAPPING {
  container network-mapping {
    description "Set of shared networks in VTC";
    tailf:info "Shared Networks";
    list network {
      key id;
      leaf id {
        description "Identifier of the shared network.";
        tailf:info "ID of the shared network.";
        type vts-types:uuid;
      } //leaf id

      leaf tenant-name {
        description "tenant name of the network";
        tailf:info "tenant name of the network";
        type vts-types:string15; // same type as tenant name
      } // leaf tenant-name

      leaf topology-id {
        description "topology id of the network";
        tailf:info "topology id of the network";
        type vts-types:string128;
      } // leaf topology-id
```

```
container ports {
  list port {
    description "List of ports configured on this shared network";
    tailf:info "List of ports configured on this shared network";

    key id;
    leaf id {
      description "Port id";
      tailf:info "port id";
      type vts-types:uuid;
    } // leaf id

    leaf tenant-name {
      description "tenant name of the port";
      tailf:info "tenant name of the port";
      type vts-types:string15; // same type as tenant name
    } // leaf tenant-name

    leaf topology-id {
      description "topology id of the port";
      tailf:info "topology id of the port";
      type vts-types:string128;
    } // leaf topology-id
  } // list port
} // container ports

container port-macs {
  list port-mac {
    key mac-address;

    leaf mac-address {
      description "Port MAC address.";
      tailf:info "MAC address.";
      type vts-types:string128;
    }

    leaf port-id {
      description "Port id";
      tailf:info "port id";
      type vts-types:uuid;
    } // leaf id
  }
} // container port-macs

container subnetworks {
  list subnetwork {
    description "List of subnets added on this shared network";
    tailf:info "List of subnets added on this shared network";

    key id;
```

```
            leaf id {
              description "subnet id";
              tailf:info "subnet id";
              type vts-types:uuid;
            } // leaf id

            container router {
              description "Router that subnet is attached to";
              tailf:info "Router that subnet is attached to";

              leaf id {
                description "Router id that subnet is attached to";
                tailf:info "Router id";
                type vts-types:uuid;
              } // leaf id

              leaf tenant-name {
                description "tenant name of the router";
                tailf:info "tenant name of the router";
                type vts-types:string15;
              } // leaf tenant-name

              leaf topology-id {
                description "topology id of the router";
                tailf:info "topology id of the router";
                type vts-types:string128;
              } // leaf topology-id
            } // container router
          } // list port
        } // container ports
      } // list shared-network
    } // container shared-networks
} // grouping VTS_SHARED_NETWORKS

grouping VTS-DEVICE-GROUP {
  container device-groups {
    description "List of device groups";
    tailf:info "List of device groups";

    list device-group {
      description "Set of devices";
      tailf:info "Set of devices";
      key name;

      leaf name {
        description "Name of the device-group";
        tailf:info "Name of the device-group";
        type vts-types:string128;
        must "0 = count(/ncs:devices/ncs:device[ncs:name = current()/../name])" {
          error-message "Device Group name could not be the same as the device
```

```
name";
          tailf:dependency "/ncs:devices/ncs:device/ncs:name";
        }
      }

      leaf type {
        description "Type of device group";
        tailf:info "Type of device group";
        type identityref {
          base "vts-ids:device-group-type";
        }
      }

      leaf vmmid {
        description "ID of VMM for device group";
        tailf:info "ID of VMM for device group";
        type vts-types:string128;
      }

      container devices {
        description "List of device in the device group";
        tailf:info "List of device in the device group";

        list device {
          description "device in the device group";
          tailf:info "device in the device group";
          key name;
          ncs:servicepoint vts-device-group-servicepoint;
          uses ncs:service-data;

          leaf name {
            description "Name of the device";
            tailf:info "Name of the device";
            type vts-types:string128;
            must "/ncs:devices/ncs:device[ncs:name = current()] or " +
            "/vts:cisco-vts/vts:vtfs/vts:vtf[ip = current()]" {
              error-message "deleting a device or vtf is not allowed when it is present in a
device group OR
                        adding a device or vtf in a device group is allowed only when it is
present in the inventory";
              tailf:dependency "/ncs:devices/ncs:device/ncs:name";
              tailf:dependency "/vts:cisco-vts/vts:vtfs/vts:vtf/vts:ip";
              tailf:dependency ".";
            }
          }

          leaf type {
            description "Type of device";
            tailf:info "Type of device";
            type identityref {
```

```
            base "vts-ids:device-group-device-type";
           }
          }

          leaf devicegroup-device-creation-timestamp {
           type ietf-yang:date-and-time;
           description "The timestamp when the device group device creation was
triggered";
           tailf:info  "The timestamp when the device group device creation was
triggered";
          }

          leaf disable-validations {
           type empty;
           description "Disable ports and admin domain validations.";
           tailf:info "Disable ports and admin domain validations.";
          }

        } //list device
      } // container devices
    } // list device-group
  } // container device-groups
 } // grouping VTS-DEVICE-GROUP

 grouping VTS-DEVICE-INTERFACE-GROUP {
  container device-interface-groups {
    description "List of device Interface groups";
    tailf:info "List of device Interface groups";

    list device-interface-group {
      description "Set of device interfaces";
      tailf:info "Set of device interfaces";
      key name;
      ncs:servicepoint vts-device-interface-group-servicepoint;
      uses ncs:service-data;

      leaf name {
        description "Name of the device-interface-group";
        tailf:info "Name of the device-interface-group";
        type vts-types:string128;
      }

      leaf type {
        description "Type of device group";
        tailf:info "Type of device group";
        type identityref {
          base "vts-ids:device-group-type";
        }
        default vts-ids:device-interface-group;
      }
```

```
      leaf is-user-defined-group {
        description "Flag to determine if it is user defined interface group";
        tailf:info "Flag to determine if it is user defined interface group";
        type boolean;
        default "false";
      }
      leaf interface-pool-change-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when poolname change was retriggered";
        tailf:info  "The timestamp when poolname changes was retriggered";
      }
      container device-interfaces {
        description "List of device interfaces in the device interface group";
        tailf:info "List of device interfaces in the device interface group";

        list device-interface {
          description "device in the device interface group";
          tailf:info "device in the device interface group";
          key "name device-name";

          leaf name {
            description "Name of the device interface";
            tailf:info "Name of the device interface";
            type vts-types:string128;
          }

          leaf device-name {
            description "Name of the device interface is associated to";
            tailf:info "Name of the device interface is associated to";
            type vts-types:string128;
          }
          leaf pool-name {
            description "Device Interface Pool Name.";
            tailf:info "Device Interface Pool Name.";
            type vts-types:string128;
          }
          leaf service-instance-id {
            description "Service Instance Id";
            tailf:info "Service Instance Id.";
            type vts-types:instance-id;
            default 0;
          }
        } //list device-interface
      } // container device-interfaces
    } // list device-interface-group
  } // container device-interface-groups
} // grouping VTS-DEVICE-INTERFACE-GROUP

grouping BONDED-INTERFACE {
  container bond {
```

```
    description "VTF vhost bonded interface.";
    tailf:info "VTF vhost bonded interface.";
    presence "interface is a bond";

    leaf bond-mode {
      description "NIC bond mode.";
      tailf:info "NIC bond mode.";
      type identityref {
        base "vts-ids:nic-bond-mode";
      }
      default vts-ids:balance-xor-l34;
    }
    container bond-interfaces {
      description "Set of interfaces that forms this bond.";
      tailf:info "Set of interfaces that forms this bond.";

      list bond-interface {
        description "List of interfaces that forms this bond.";
        tailf:info "List of interfaces that forms this bond.";
        key name;
        min-elements 1;

        leaf name {
          description "Name of interface on the compute host.";
          tailf:info "Name of interface on the compute host.";
          type string;
        }
      } // list bond-interface
    } // container bond-interfaces
  } // container bond
 } // grouping BONDED-INTERFACE
}
```

6.9 System Config

```
submodule cisco-vts-system-config {

 belongs-to cisco-vts {
   prefix vts;
 }

 import ietf-inet-types {
   prefix inet;
 }
 import ietf-yang-types {
   prefix ietf-yang;
 }

 import tailf-common {
```

```
  prefix tailf;
}

import tailf-ncs {
  prefix ncs;
}

import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix vts-ids;
}


organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the
   management of system level configurations.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2017-05-30" {
  description
    "VTS 2.5.1
     Enhancements:
      Added vtf-mode-config container to determine VTF usage.
   ";
}

revision "2016-11-15" {
  description
    "VTS 2.4
     Enhancements:
     Corrections:
        - The ip address of DHCP servers can also be an IPv6 address
        - Added constraints so that only one v4 and one v6 DHCP server
          are allowed.
```

```
        - Added a default for the DHCP server vrf-name, which is always required.
    ";
  }

  revision "2016-08-30" {
    description
      "VTS 2.3.1
       Enhancements:
          - Added leaf domain-id to identify this domains routes
      - Added leaf gwgroup-change-timestamp to track RR gateway group changes
       Corrections:
    ";
  }

  revision "2016-07-15" {
    description
      "VTS 2.3.0
       Enhancements:
        - Added multicast-config leaf to control the number of
            networks per multicast group ip allocation
       Corrections:
    ";

  }


  revision "2015-06-15" {
   description
      "Initial revision.";
  }


  grouping SYSTEM-CONFIG {
    container global-settings {
      description "Container for system level configurations";

      leaf route-reflector-mode {
        //TODO: improve this description
        description "Determines how the route reflector operate.";
        tailf:info "Route reflector mode.";
        type identityref {
          base "vts-ids:route-reflector-mode";
        }
      }
      container global-route-reflectors {
        description "Set of global route reflectors.";
        tailf:info "Global route reflectors.";
        when "../route-reflector-mode = 'vts-ids:global-rr'";
        list global-route-reflector {
          uses ncs:service-data;
```

```
        ncs:servicepoint global-rr-servicepoint;
        description "List of global route reflectors.";
        tailf:info "Route reflector.";
        key name;
        leaf name {
          description "Name of the route reflector.";
          tailf:info "Name.";
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
        }
        leaf gwgroup-change-timestamp {
          type ietf-yang:date-and-time;
          description "The timestamp when the gateway group change was triggered";
          tailf:info  "The timestamp when the gateway group change was triggered";
        } // leaf gwgroup-change-timestamp
      } // list route-reflector
    } // container route-reflectors

    container dhcp-servers {
      description "Set of DHCP servers.";
      tailf:info "DHCP servers.";
      list dhcp-server {
        uses ncs:service-data;
        description "List of DHCP servers.";
        tailf:info "DHCP server.";
        key ip-address;

        /* TODO: enforce that only 1 v4 and 1 v6 can be taken
              the constraints below are not enforcing that.
              Revisit this ASAP

        must "count(contains(../dhcp-server/ip-address, ':')) <= 1" {
          error-message "No more than one DHCP server with an IPv6 address can be
provided.";
          tailf:dependency ".";
        }

        must "count(contains(../dhcp-server/ip-address, '.')) <= 1" {
          error-message "No more than one DHCP server with an IPv4 address can be
provided.";
          tailf:dependency ".";
        }
        */


        leaf ip-address {
          description "IP address of the DHCP server";
          tailf:info "IP address.";
          type inet:ip-address;
```

```
        }

        leaf vrf-name {
          description "VRF name of the DHCP server";
          tailf:info "VRF name.";
          type vts-types:string128;
          default "default";
        }

      } // list dhcp-server
    } // container dhcp-servers

    container anycast-gateway {
      description "Common anycast configuration for all gateways.";
      tailf:info "Common anycast gateway.";
      presence "Existence of anycast-gateway configuration";
      uses ncs:service-data;
      ncs:servicepoint vts-anycast-gateway-servicepoint;
      leaf anycast-gw-address {
        description "Common anycast MAC address for all gateways.";
        tailf:info "Common anycast MAC address.";
        type string {
          pattern '[0-9a-fA-F][02468aceACE](:[0-9a-fA-F]{2}){5}' {
            error-message "The anycast gateway mac address must be a unicast
address.";
          }
        }
      }
    } // container anycast-gateway

    leaf domain-id {
       type string;
       tailf:info "Id/Tag to identify this domains routes";
    }

    container service-policy-config{
      description "Service Policy Configuration";
      tailf:info "Service Policy Configuration";
      leaf policy-mode {
        description "Specify service policy mode";
        tailf:info "Specify service policy mode";
        default "vts-ids:no-policy-mode";
        type identityref {
          base "vts-ids:service-policy-mode";
        }
      }
    }//container service-policy-config

    container multicast-config {
      description "Multicast config";
```

```
        tailf:info "Multicast config";
        leaf allocation-reuse-count {
          description "Number of networks per multicast group ip allocation.  A value of
zero indicates that all networks in the system use the same multicast group";
          tailf:info "Number of networks per multicast group ip allocation.  A value of
zero indicates that all networks in the system use the same multicast group";
          type vts-types:multicast-reuse-count;
          //type uint32;
          default 1;
        }
      }// container multicast-config

      container resource-allocator-config {
        description "Resource Allocator config";
        tailf:info "Resource Allocator config";
        leaf allocate-outside-range {
          description "Allow allocations to occur outside of any range?";
          tailf:info "Allow allocations to occur outside of any range?";
          type boolean;
          default true;
        }
      }// container resource-allocator-config


      container delay-reading-applied-config {
        description "Time to wait to read the running configuration of devices after it is
applied.";
        tailf:info "Time to wait to read the running configuration of devices  after it is
applied.";
        presence "Existence of delay-reading configuration";
        uses ncs:service-data;
        ncs:servicepoint vts-read-delay-servicepoint;
        leaf nxos-delay-reading-applied-config {
          description "Time to wait to read the running configuration after it is applied
for NX-OS devices.";
          tailf:info "Time to wait to read the running configuration after it is applied NX-
OS devices.";
          type uint16;
          units milliseconds;
          default 0;
        }  // leaf nxos-delay-reading-applied-config
      } // container delay-reading-applied-config

      container southbound-lock-config {
        description "Southbound lock config";
        tailf:info "Southbound lock config";
        leaf device-monitoring {
          description "Southbounds locks a device if down and syncs missing data when
it comes up again";
          tailf:info "Southbounds locks a device if down and syncs missing data when it
```

```
comes up again";
          type boolean;
          default true;
        }
      }// container southbound-lock-config

      container vtf-mode-config {
        description "VTF Mode config";
        tailf:info "VTF Mode config";
        leaf vtf-mode {
          description "VTF mode indicates whether VTC considers VTFs as L2 or
VTEP.";
          tailf:info "VTF mode indicates whether VTC considers VTFs as L2 or VTEP.";
          type vts-types:vtf-mode-enum;
          default l2;
        }
      }
      container multiple-subnets-allowed {
        description "Flag to support multiple subnets";
        leaf multiple-subnets-allowed {
          description "Flag to support multiple subnets";
          tailf:info "Flag to support multiple subnets";
          type boolean;
          default "false";
         }
      }

    } // container system-config
  } // grouping
} // module
```