NSO NFVO Bundle

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Introduction to the NFVO Bundle

Introduction

This document describes how to use NSO and the Elastic Services Controller (ESC). Mapped to the ETSI NFV Architecture NSO performs the OSS and NFVO functionality and the ESC implements the VNFM functionality.

Please note, **it is assumed the reader is familiar with the ETSI MANO nomenclature.**

NSO and ESC communicates over a YANG/NETCONF interface (Nfvo-Vnfm).

ESC manages on-boarding, instantiation and monitoring of the VMs.

Figure 1. The ETSI NFV Architecture

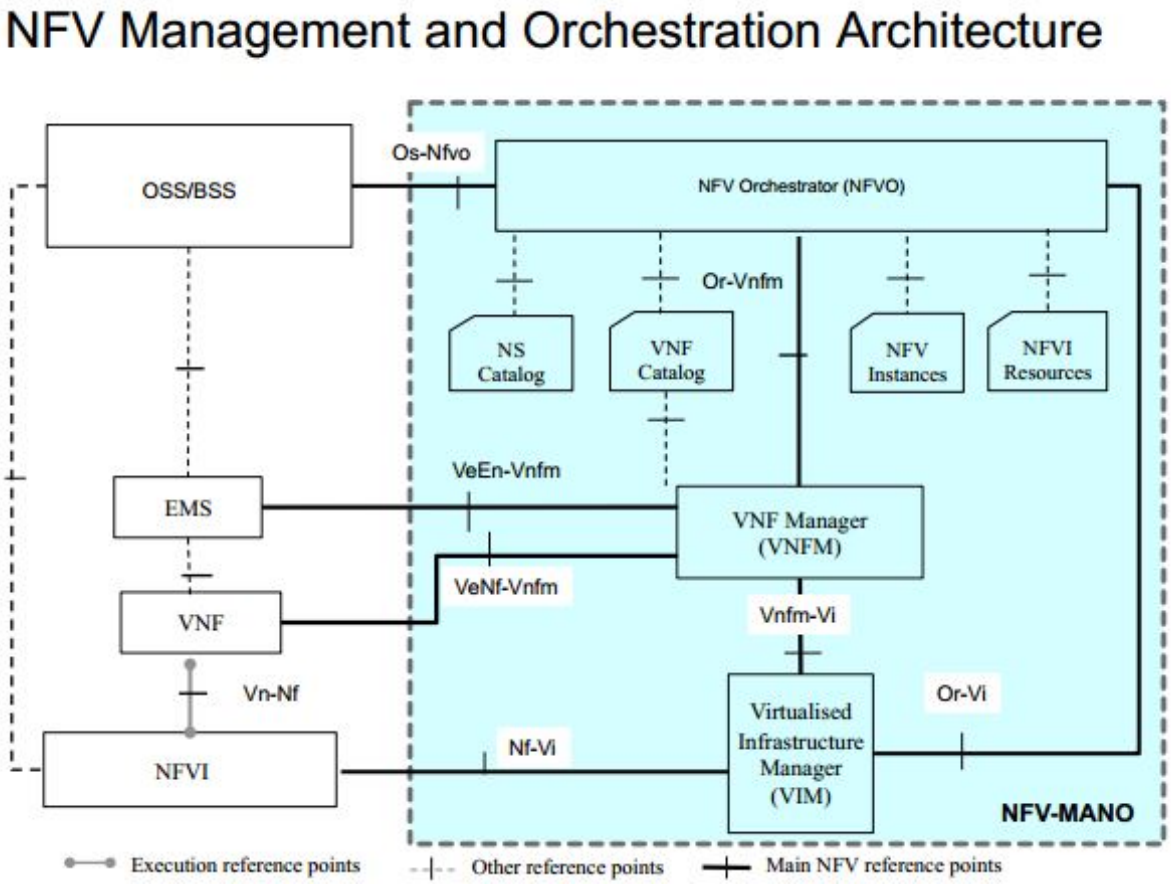
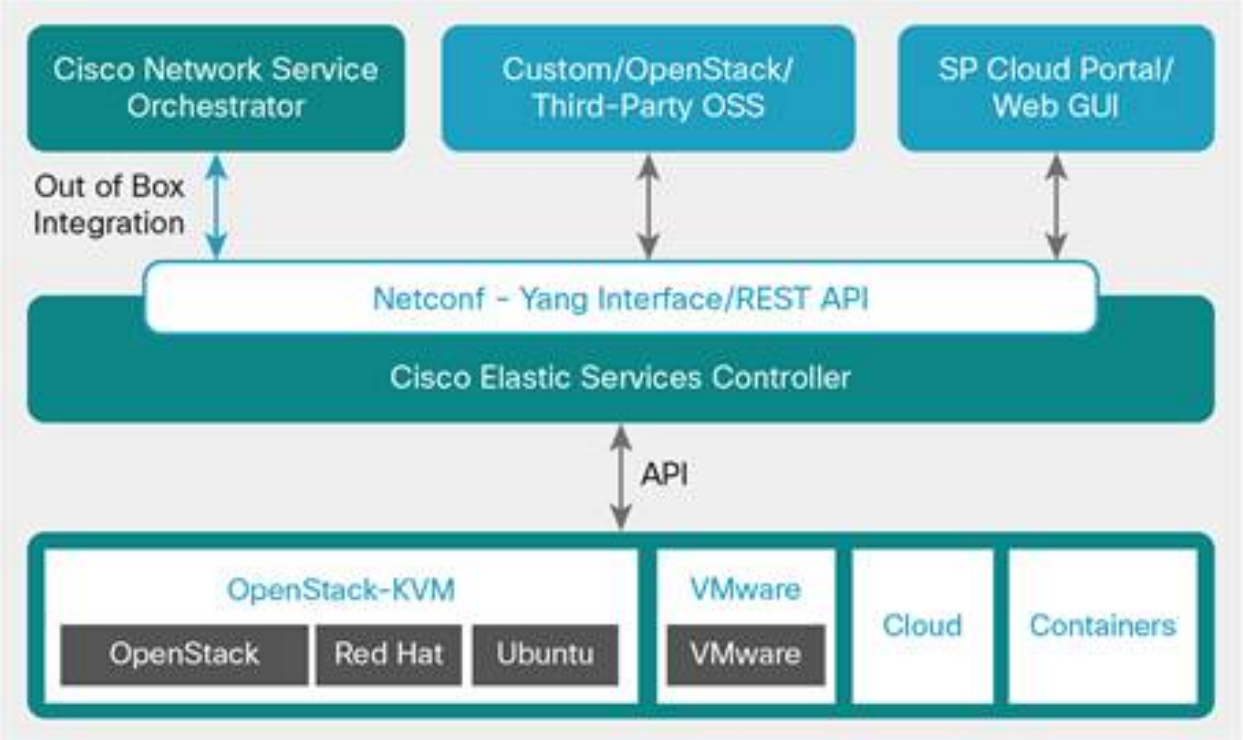


Figure 2. ESC Architecture



Overview

Cisco Elastic Services Controller (ESC) is a Virtual Network Functions Manager (VNFM), performing life cycle management of Virtual Network Functions (VNFs). ESC provides agent-less and multi vendor VNF management by provisioning the virtual services, and monitoring their health and load. ESC provides the flexibility to define rules for monitoring, and associate actions to be triggered based on the outcome of these rules. Based on the monitoring results, ESC performs scale in or scale out on the VNFs. It also supports automatic VM recovery when a VM fails.

NSO manages the end-customer specific service configuration on top of the running virtual machines. This covers both the NFVO and the OSS part. An example is that ESC manages to start a virtual PE and its initial configuration and then NSO will create a customer specific VPN over that vPE.

There is a bit of terminology confusion looking at the two roles NSO and ESC takes. ETSI MANO uses the term network service to refer to a set of started VNFs that are stitched together. In the telco world the term service is more generally referring to end-customer specific service instances possible running over VMs.

ESC manages the ETSI definition of a service: a group of managed VMs. NSO manages the telco oriented definition of a service and also delegates the need for any ETSI MANO services to ESC. So in a Cloud VPN use case for example: the self-service portal requests NSO to create a VPN. NSO might detect that a virtual PE is needed in a data-center, NSO will then request ESC to start that virtual PE as part of the Service Orchestration. There are several benefits of managing the complete orchestration request as one transaction via NSO:

- The complete service (VPN service and MANO vPE service) is managed in one YANG model and one transaction.

- Any life-cycle changes can impact the two service contexts: a healing event in MANO need to affect the VPN service and vice versa.

You could choose to keep the MANO part separate from the OSS part. In that case NSO/ESC would start a virtual PE and consider that part done. Then as a second step the service configuration is performed. But then you loose all the benefits above.

So the way to best utilize NSO and ESC together is to view Virtual Machines as a result from a service activation request in the OSS layer. NSO manages the complete end-customer service activation requests which covers physical devices, other support systems and often virtual network functions, all in one piece.

ESC Concepts

- Onboarding: this corresponds to VNF Package on-boarding. In order for ESC to start a VM it needs the image and OpenStack flavor information, ETSI VNFD. You can also register VNFs, a VNF with different VMs. Registration corresponds to the "service" part of the ESC YANG data-model.
- Instantiation: an onboarded VNF can be instantiated/deployed which makes the VMs running. Instantiation includes day0 configuration, affinity policies etc. Instantiation corresponds to the "tenant" part of the ESC data-model.
- Notifications: ESC sends NETCONF notifications that NSO needs to react to and manage in the Reactive FASTMAP loop. At VNF deployment ESC sends a VM_ALIVE notification when the VM is started. There are also scaling and healing events that need to trigger Reactive FASTMAP.

Make sure you read the "Cisco Elastic Services Controller User Guide" to have a good understanding of all these concepts before continuing.

NSO Concepts

A "service" in NSO has a different meaning than in the ESC data-model and in the ETSI NFV MANO specification. In NSO a service refers to run-time configuration for a specific service instance for a specific user. This often spans physical and virtual devices. The NSO service-model defines the input parameters for the service. In the ETSI MANO specification this corresponds to the OSS layer. The mapping from the input parameters defined by the service model to the network is defined in NSO mapping logic. Mapping logic can be Java/Python or configuration templates, most often a combination of both. What is important to understand is that this mapping includes registering and deploying a VNF as a "sub-step". VMs are needed since we want to provision an end-to-end service.

NSO uses state-based convergence for the mapping logic, NSO moves the service to its final desired state, fully provisioned including started fully configured VMs. The algorithm behind the scenes is called Reactive FASTMAP, RFM. The basic modus operandi of RFM in virtual use cases is that NSO at first iteration realizes that VMs need to be started; NSO registers the VNF and requests ESC to deploy it, at this point NSO exits the first iteration in the state convergence process. When ESC later sends the VM_ALIVE notification the service is "redeployed" which triggers the state convergence for the service again, at this point the state convergence passes the VM started state and probably continues with VNF run-time configuration. Several different events might trigger further service redeploy operations. For example scaling and healing events need to trigger redeploy so that NSO can configure, reconfigure, remove configuration etc based on the current state in the virtual infra-structure.

Make sure you read the section "Developing NCS Services" in the "NSO Development Guide". Pay special attention to subsections at the end on Reactive FASTMAP and virtual devices.

NFVO Bundle Architecture

The NFVO Bundle (the Bundle) enables use of MANO compliant VNFDs and NSDs in NSO. It also enables onboarding and and instantiation of these on ESC. Further, if the VNF are to be managed by NSO, the Bundle handles device creation/deletion in the NSO device tree.

Descriptors and Records

In the MANO context a *descriptor* describes the metadata for a VNF or NS and supplies the general data describing these, examples are number of virtual CPUs, memory requirements etc. A *record* on the other hand describes the runtime data needed to instantiate a VNF or a NS. This can for example be IP addresses, the actual network names which the VNF connects to, etc.

Packages

The Bundle consists of a number of packages but the two core packages are tailf-nfvo and tailf-nfvo-esc. Please see the section called “Model Overview” for the actual YANG models.

tailf-nfvo

This package contains the YANG models according to the MANO specification. Most important the schema defines all the descriptors in MANO like NSD, VNFD etc.

tailf-nfvo-esc

tailf-nfvo-esc implements the MANO records for instantiaion on ESC and OpenStack, i.e. the MANO records, e.g. VNFR and NSR.

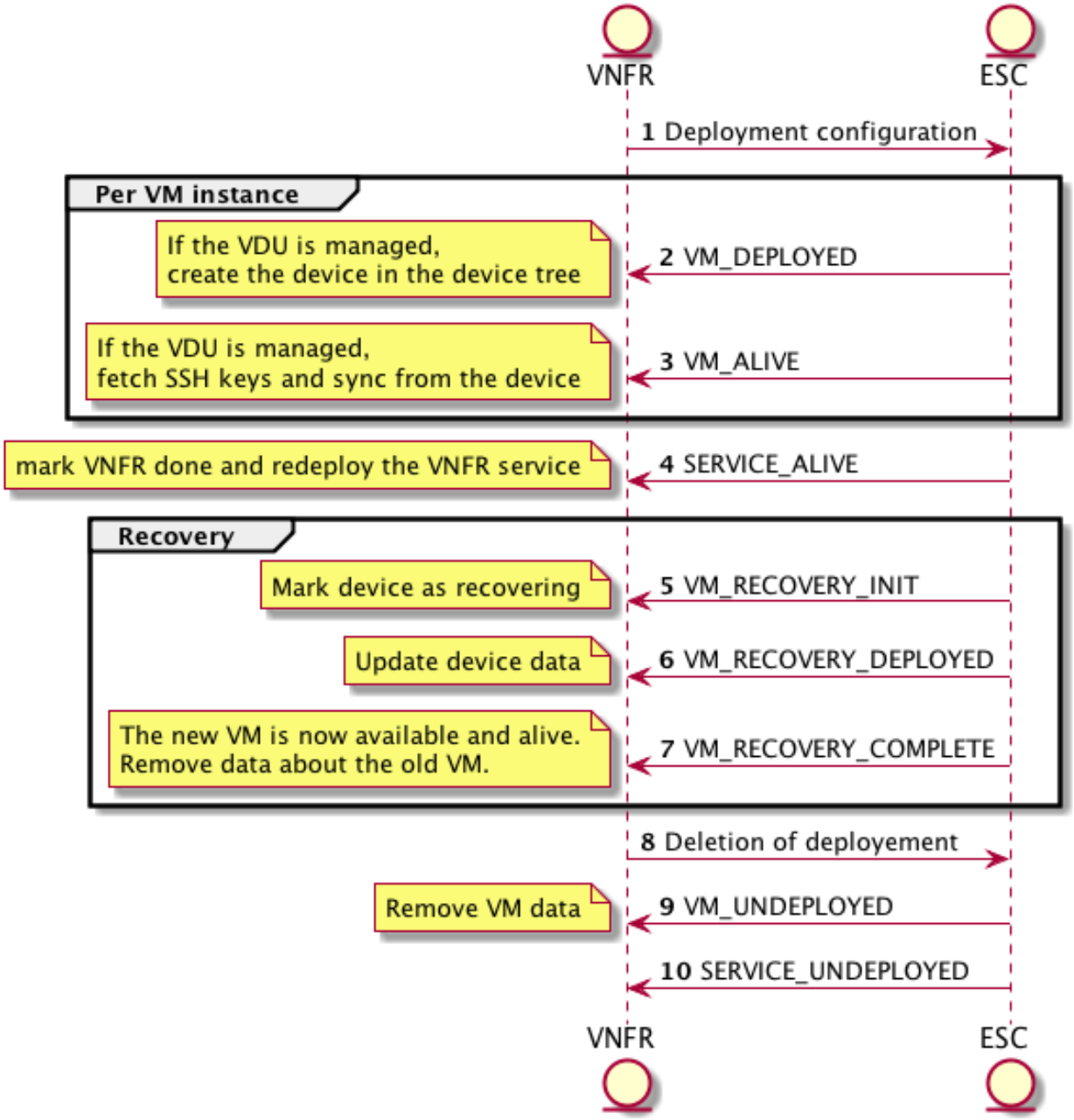
The main services in this packet are the VNFR and the NSR services. Any northbound users will interact with these services to start VNFs or network services. Please see Chapter 4, *Developing for the NFVO Bundle* for examples on how to develop these services.

VNFR and ESC interaction

This section describes the interaction between the VNFR service (the Service) in NSO and ESC.

The Service writes configuration to ESC using the ESC NETCONF NED. ESC signals results and events back to the Service through NETCONF notifications. Please see Figure 3, “VNFR/ESC Interaction” for the message flow.

Figure 3. VNFR/ESC Interaction



1. The VNFR service writes configuration data to ESC for one or more VDUs.
2. When OpenStack has deployed the VM, ESC sends VM_DEPLOYED. The VNFR service creates the device and marks it as deploying.
3. When ESC can contact the VM it sends VM_ALIVE. The VNFR service marks the device as ready. If the device is managed, the service sets up SSH and syncs from the device.
4. When all VMs in the deployment are alive, ESC sends SERVICE_ALIVE. The VNFR service marks itself as ready and redeploys northbound services.

5. In case ESC looses connection with the VM, ESC will send VM_RECOVERY_INIT to indicate it is trying to recover the VM. The VNFR service marks the device as recovering.
6. If the VM is rebuilt from scratch with only day0 configuration, ESC sends VM_RECOVERY_DEPLOYED. The VNFR service updates the device table with the new device and redeploys northbound services to allow re-application of day-n configuration.
7. When the VM is recovered, ESC sends VM_RECOVERY_COMPLETE. The VNFR service redeploys northbound services to allow them to reapply day-n configuration.
8. The deployment configuration is deleted from ESC.
9. For each VM, ESC send VM_UNDEPLOYED. The VNFR service removes data from the device table.
10. When the entire deployment is deleted, ESC sends SERVICE_UNDEPLOYED.

CHAPTER 2

Installing the NFVO Bundle

Requirements

- NSO 4.2+
- ESC 2.1.5.18

Installing the NFVO Bundle

The packages are distributed as an NSO bundle. This bundle contains all packages and can be installed in a NSO project directory.

```
$ tar zxf nfvo-1.0.tar.gz
```

The *packages* directory should now contain the following packages

- esc
- tailf-nfvo
- tailf-nfvo-esc
- tailf-nfvo-examples
- tailf-nfvo-staged-delete-example
- tailf-nfvo-ui

The *tailf-nfvo-examples* can safely be removed if not needed. The *tailf-nfvo-staged-delete-example* package shows how to implement staged delete, this package can be removed if no staged delete functionality is needed.

After installing the bundle in a running system, the packages has to be reloaded. This can be done in the CLI.

```
$ ncs_cli -u admin -C
admin@ncs# packages reload
reload-result {
    package esc
```

```
    result true
}
reload-result {
    package tailf-nfvo
    result true
}
reload-result {
    package tailf-nfvo-esc
    result true
}
reload-result {
    package tailf-nfvo-examples
    result true
}
reload-result {
    package tailf-nfvo-staged-delete-example
    result true
}
```

Building the ESC NED

ESC supports a northbound NETCONF/YANG interface. This means that NSO can talk native NETCONF towards ESC. This section walks you through the required steps to build a ESC NED.

Please NOTE: you only need to build an ESC NED if you have a installed a later version of ESC than what is shipped in the NFVO Bundle.

The ESC YANG files can be found in the ESC installation directory at `esc-confd/YANGmodels-tailf`.

Procedure 2.1. Building the ESC NED

Step 1 Prepare a directory with the ESC YANG files

Step 2 Run ncs-make-package to build a NETCONF NED based on these YANG files.

Step 3 **make** the NED package

Step 4 Edit the metadata file for the generated NED

These steps are described below.

Prepare a directory with the ESC YANG files somewhere, say `~/esc/esc-yang`. It should look something like this:

```
$ls -l
-rw-r--r--  1 svallin  staff  5491 Jun  9 14:31 esc.yang
-rw-r--r--  1 svallin  staff  3565 Jun  9 14:31 esc_config_data.yang
-rw-r--r--  1 svallin  staff  5411 Jun  9 14:31 esc_datamodel.yang
-rw-r--r--  1 svallin  staff  2285 Jun  9 14:31 esc_dependencies.yang
-rw-r--r--  1 svallin  staff  3088 Jun  9 14:31 esc_disk.yang
-rw-r--r--  1 svallin  staff  2581 Jun  9 14:31 esc_flavor.yang
-rw-r--r--  1 svallin  staff  3209 Jun  9 14:31 esc_interface.yang
-rw-r--r--  1 svallin  staff  6085 Jun  9 14:31 esc_kpi.yang
-rw-r--r--  1 svallin  staff  2989 Jun  9 14:31 esc_network.yang
-rw-r--r--  1 svallin  staff  4286 Jun  9 14:31 esc_notifications.yang
-rw-r--r--  1 svallin  staff   920 Jun  9 14:31 esc_opdata.yang
-rw-r--r--  1 svallin  staff  7101 Jun  9 14:31 esc_opdata_devstats.yang
-rw-r--r--  1 svallin  staff  1685 Jun  9 14:31 esc_opdata_interface.yang
-rw-r--r--  1 svallin  staff  2255 Jun  9 14:31 esc_opdata_networks.yang
```

```
-rw-r--r--  1 svallin  staff  1166 Jun  9 14:31 esc_opdata_state_machines.yang
-rw-r--r--  1 svallin  staff  3627 Jun  9 14:31 esc_opdata_tenant.yang
-rw-r--r--  1 svallin  staff  6495 Jun  9 14:31 esc_policies.yang
-rw-r--r--  1 svallin  staff  2259 Jun  9 14:31 esc_rules.yang
-rw-r--r--  1 svallin  staff  5230 Jun  9 14:31 esc_scaling.yang
-rw-r--r--  1 svallin  staff  9684 Jun  9 14:31 esc_types.yang
-rw-r--r--  1 svallin  staff  3579 Jun  9 14:31 esc_volume.yang
```

Next step is to build the package based on these files. Create another directory where the NED Package will be created:

```
$ mkdir ~/esc/esc-ned
$ cd ~/esc
```

Now create the NED package like this:

```
$ ncs-make-package --netconf-ned esc-yang --dest esc-ned esc
```

The command above does:

- Create a NETCONF NED
- Store the resulting NED Package in `esc-ned`.
- Read YANG files from `esc-yang`.
- Name the package ned (the last argument).

The final step is now to **make** the ESC NED package. This step builds the necessary load files (fxs) for NSO based on the generated package. In order to do this change directory to the `src` directory in the generated NED. And then type **make** to build the package.

```
$ cd esc-ned/src/
$ make

...
/Users/svallin/dev/trunk/ncs_dir/netsim/confd/bin/confdc --yangpath ../src/yang \
          `ls ../src/yang/esc_volume-ann.yang  > /dev/null 2>  \
              echo "-a ../src/yang/esc_volume-ann.yang"` \
          -c -o esc_volume.fxs  ../src/yang/esc_volume.yang
```

The final step is to modify the metadata-file for the NED. This file is located in the root directory of the generated NED.

```
$ less package-meta-data.xml

<ncs-package xmlns="http://tail-f.com/ns/ncs-packages">
  <name>esc</name>
  <package-version>1.0</package-version>
  <description>Generated NETCONF package</description>
  <ncs-min-version>2.2</ncs-min-version>
  <component>
    <name>esc</name>
    <ned>
      <netconf/>
      <device>
        <vendor>Acme</vendor>
      </device>
```

```
    </ned>
  </component>
</ncs-package></programlisting>
```

Modify that file in your favorite editor and change the fields. This is just for display purposes and has no impact on functionality

Using the ESC NED in NSO

The above described process generates a NED package for NSO. Add that package to your NSO installation. In your NSO run-time directory add it to the `packages` directory. Then request NSO to reload packages or restart NSO with option *with-package-reload*.

```
# reload packages
```

```
$ ncs --with-package-reload
```

CHAPTER 3

Using the NFVO Bundle

Introduction

This chapter gives an overview of the different descriptors in the NFVO Bundle.

NFVO Configuration

The NFVO ESC Bundle requires some configuration in order to work.

```
# show full-configuration nfvo settings-esc
nfvo settings-esc netconf-subscription username admin
nfvo settings-esc netconf-subscription esc-device esc0
!
nfvo settings-esc netconf-subscription esc-device esc1
!
```

The netconf-subscriptions settings are mandatory. The username must be authorized to setup a NETCONF subscription to the ESC devices.

Working with VLDs

The VLD has no other function than act as a reference point for VLRs.

Working with VLRs

The VLRs are used to reference networks in the VIM. By creating a VLR with an id matching the name of the VIM network, that network can be referenced when instantiating VNFDs and NSDs.

For example a network named "office-net" is represented by

```
# show full-configuration nfvo vlr esc vlr
vlr office-net
```

Working with VNFDs

A VNF is described using a descriptor format according to Example 15, “VNF YANG Models”. Examples of VNFDs can be seen here Example 5, “VNFD Catalogue”.

The VNFD should contain all information needed to boot virtual machines based on this VNFD. It should also define it's network connection points.

VDU

Each VNFD consists of one or more Virtual Deployment Units. A VDU corresponds one virtual machine and it contains the data describing that specific VM type.

For each VDU; CPU, memory and storage requirements has to be specified. It's also possible to specify a image URL here. This URL can be used to onboard the image in OpenStack.

A VDU specifies one virtual machine and each VDU contains at least one connection point.

VDU Connection Point

A VDU's connection point has to connect to an internal virtual link or and external connection point.

Device Type

If the VDU can be managed by NSO, it's possible to specify NED information here.

Internal Virtual Link

A VNFD can contain one or more internal virtual links.

VDU or VNFD connection points can connect to these virtual links.

External Connection Point

This section defines the network connection points this VNFD exposed to applications.

An external connection point can be marked as the management interface. If this VNFD is managed by NSO, NSO will try to connect to this interface(s).

Deployment Flavor

The deployment flavor and instantiation levels specify which and how many of each VDU to instantiate.

The VDU profiles describes minimum and maximum number of VDU instances in the flavor. The initial number of instances is set in the instantiation level(s).

Working with VNFRs

The VNFR contains all data needed to instantiate a VNFD on the VIM.

NOTE! You have to create the entire VNFR **before comitting**, otherwise you risk having ESC trying to start a incomplete deployment.

Instantiation

```
# show full-configuration nfvo vnfr
nfvo vnfr esc vnf-deployment volvo branch-office
 username admin
 esc-device esc0
 !
 vnfr CSR1kv
  vnfd-flavor         basic
  instantiation-level basic
  managed
  vdu CSR
   image-name  CSR9.3.14.1
   flavor-name CSR9.3.14.1
   day0 iosxe_config.txt
    url http://10.147.46.245/nfvo-modelling/CSR_day0.txt
    variable ADMIN_PWD
     value cisco123
    !
    variable HOSTNAME
     value the-router
    !
   !
   authgroup   csr
   connection-point-address mgmt
    start 10.1.1.20
    end   10.1.1.30
   !
  !
  vnfd-connection-point left
   vlr internet
  !
  vnfd-connection-point mgmt
   vlr mgmt
  !
  vnfd-connection-point right
   vlr volvo-internal
  !
 !
!
```

This will create an instance (branch-office) of a CSR for a tenant (volvo).

The VDU will be *managed* and using VIM specified VIM image (CSRImage) and flavor (CSRFlavor), see the section called “Onboarding flavor and image to the VIM”.

For each day0 file, it's possible to specify the source URL and key/value pairs that will be substituted in the day0 data. In this example *ADMIN_PWD* and *HOSTNAME* are sample day0 key value pairs provided.

For a managed device, the authgroup used to connect to this device has to be specified.

The connection points specified in the VNFD are mapped to VIM networks using VLRs. In this example the CSR will connect to internet, p_net and volvo-internal. These networks are assumed to exist in the VIM.

The instantiation of a VNF results in configuration in ESC.

```
# show full-configuration devices device esc0 config esc:esc_datamodel tenants tenant volvo depl
vm_group CSR {
    image              CSR9.3.14.1;
    flavor             CSR9.3.14.1;
    bootup_time        300;
    recovery_wait_time 120;
    interfaces {
        interface 0 {
            network    mgmt;
            ip_address 10.1.1.20;
        }
        interface 1 {
            network internet;
        }
        interface 2 {
            network volvo-internal;
        }
    }
    kpi_data {
        kpi VM_ALIVE {
            metric_value            1;
            metric_cond             GT;
            metric_type             UINT32;
            metric_occurrences_true  1;
            metric_occurrences_false 5;
            metric_collector {
                type             ICMPPing;
                nicid            0;
                poll_frequency   5;
                polling_unit     seconds;
                continuous_alarm false;
            }
        }
    }
    rules {
        admin_rules {
            rule VM_ALIVE {
                action [ "ALWAYS log" "FALSE recover autohealing" "TRUE servicebooted.sh" ];
            }
        }
    }
    config_data {
        configuration iosxe_config.txt {
            file http://10.147.46.245/nfvo-modelling/CSR_day0.txt;
            variable ADMIN_PWD {
                val [ cisco123 ];
            }
            variable HOSTNAME {
                val [ the-router ];
            }
        }
    }
    scaling {
        min_active 1;
        max_active 10;
        static_ip_address_pool mgmt {
            ip_address_range 10.1.1.20 10.1.1.30;
        }
    }
    extensions {
```

```
        extension NSO {
            properties {
                property AUTHGRP {
                    value default;
                }
                property MGMTIF {
                    value 0;
                }
                property NEDID {
                    value ios-id:cisco-ios;
                }
                property NEDTYPE {
                    value cli;
                }
                property VDU {
                    value CSR;
                }
                property VNFR {
                    value CSR1kv;
                }
            }
        }
    }
}
```

In the *NSO extension* configuration, we store data that is used if the VM should be added to NSO's device tree.

Setting additional parameters using device templates

In some cases it's necessary to be able to pass additional parameters to ESC, an example could be scaling parameters. This can be achieved by passing these device templates to the device-template list in the VNFR.

The device-template list is ordered by user and applied in the same order. To use the template below, you would add scale-out to the device-template list.

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <template>
      <name>scale-out</name>
      <config>
        <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
          <tenants>
            <tenant>
              <name>{$TENANT}</name>
              <deployments>
                <deployment>
                  <name>{$DEPLOYMENT_NAME}</name>
                  <vm_group>
                    <name>{$VDU}</name>
                    <kpi_data>
                      <kpi>
                        <event_name>VM_OVERLOADED</event_name>
                        <metric_value>30</metric_value>
                        <metric_cond>GT</metric_cond>
                        <metric_type>UINT32</metric_type>
                        <metric_occurrences_true>2</metric_occurrences_true>
                        <metric_occurrences_false>4</metric_occurrences_false>
                        <metric_collector>
                          <type>CPU</type>
```

```
                <nicid>0</nicid>
                <poll_frequency>3</poll_frequency>
                <polling_unit>seconds</polling_unit>
                <continuous_alarm>false</continuous_alarm>
              </metric_collector>
            </kpi>
          </kpi_data>
          <rules>
            <admin_rules>
              <rule>
                <event_name>VM_OVERLOADED</event_name>
                <action>ALWAYS log</action>
                <action>TRUE servicescaleup.sh</action>
              </rule>
            </admin_rules>
          </rules>
        </vm_group>
      </deployment>
    </deployments>
  </tenant>
 </tenants>
</esc_datamodel>
</config>
</template>
</devices>
</config>
```

Working with NSDs

A network service is described using a descriptor format according to Example 16, “NS YANG Models”. An examples of NSD can be seen here Example 6, “NSD Catalogue”.

With a NSD it's possible to describe a MANO network service consisting of several VNFDs and network service chains between these VNFDs.

VNFD

This section lists all included VNFDs. It also describes how each VNFD's connection point connects to a NSD service access point (SAP) or a NSD virtual link.

Service Access Point

The SAP defines the connection points the network service exposes to applications.

Virtual Link

A NSD can contain one or more virtual links.

SAP or VNFD connection points can connect to these virtual links.

Working with NSRs

```
# show full-configuration nfvo nsr
esc {
  nsr volvo fw-router {
    username admin;
    nsd      fw-router;
```

```
    esc-device esc0;
    vnfr ASA {
      vnfd-flavor         dep-fw;
      instantiation-level small;
      vdu firewall {
        image-name  ASAImage;
        flavor-name ASAFlavor;
        day0 day0-config {
          url http://10.147.46.245/nfvo-modelling/ASA_941_day0.txt;
          variable ADMIN_PWD {
            value cisco123;
          }
          variable HOSTNAME {
            value the-fw;
          }
        }
      }
    }
    vnfr CSR1kv {
      vnfd-flavor         basic;
      instantiation-level basic;
      vdu CSR {
        image-name  CSRImage;
        flavor-name CSRFlavor;
        day0 iosxe_config.txt {
          url http://10.147.46.245/nfvo-modelling/CSR_day0.txt;
          variable ADMIN_PWD {
            value cisco123;
          }
          variable HOSTNAME {
            value the-router;
          }
        }
      }
    }
    virtual-link inside-net {
      dhcp;
      subnet {
        network 10.0.0.0/24;
        gateway 10.0.0.1;
      }
    }
    service-access-point inside {
      vlr office-net;
    }
    service-access-point mgmt {
      vlr p_net;
    }
    service-access-point outside {
      vlr internet;
    }
  }
}
```

In this example a NSD with two VNFDs (CSRv and ASAv) is instantiated.

For each VNF flavor and instantiation level is specified. For each VDU image, flavor and day0 data is provided.

This NSR also defines the internal virtual link *inside-net*'s networking parameters.

The NSR is connected to VIM network in the same way a VNFR is.

In ESC this results in the following configuration:

```
# show full-configuration devices device esc0 config esc:esc_datamodel tenants tenant volvo depl
networks {
    network volvo-fw-router-inside-net {
        shared false;
        subnet volvo-fw-router-inside-net-subnet {
            ipversion ipv4;
            dhcp      true;
            address   10.0.0.0;
            netmask   255.255.255.0;
            gateway   10.0.0.1;
        }
    }
}
vm_group CSR {
    image              CSRImage;
    flavor             CSRFlavor;
    bootup_time        300;
    recovery_wait_time 120;
    interfaces {
        interface 0 {
            network p_net;
        }
        interface 1 {
            network volvo-fw-router-inside-net;
        }
        interface 2 {
            network office-net;
        }
    }
    kpi_data {
        kpi VM_ALIVE {
            metric_value            1;
            metric_cond             GT;
            metric_type             UINT32;
            metric_occurrences_true  1;
            metric_occurrences_false 5;
            metric_collector {
                type             ICMPPing;
                nicid            0;
                poll_frequency   5;
                polling_unit     seconds;
                continuous_alarm false;
            }
        }
    }
    rules {
        admin_rules {
            rule VM_ALIVE {
                action [ "ALWAYS log" "FALSE recover autohealing" "TRUE servicebooted.sh" ];
            }
        }
    }
    config_data {
        configuration iosxe_config.txt {
            file http://10.147.46.245/nfvo-modelling/CSR_day0.txt;
            variable ADMIN_PWD {
                val [ cisco123 ];
            }
            variable HOSTNAME {
                val [ the-router ];
```

```
            }
        }
    }
    scaling {
        min_active 1;
        max_active 10;
    }
    extensions {
        extension NSO {
            properties {
                property VDU {
                    value CSR;
                }
                property VNFR {
                    value CSR1kv;
                }
            }
        }
    }
}
vm_group firewall {
    image              ASAImage;
    flavor             ASAFlavor;
    bootup_time        300;
    recovery_wait_time 60;
    interfaces {
        interface 0 {
            network p_net;
        }
        interface 1 {
            network volvo-fw-router-inside-net;
        }
        interface 2 {
            network internet;
        }
    }
    kpi_data {
        kpi VM_ALIVE {
            metric_value            1;
            metric_cond             GT;
            metric_type             UINT32;
            metric_occurrences_true  1;
            metric_occurrences_false 5;
            metric_collector {
                type             ICMPPing;
                nicid            0;
                poll_frequency   5;
                polling_unit     seconds;
                continuous_alarm false;
            }
        }
    }
    rules {
        admin_rules {
            rule VM_ALIVE {
                action [ "ALWAYS log" "FALSE recover autohealing" "TRUE servicebooted.sh" ];
            }
        }
    }
    config_data {
        configuration day0-config {
            file http://10.147.46.245/nfvo-modelling/ASA_941_day0.txt;
```

```
            variable ADMIN_PWD {
                val [ cisco123 ];
            }
            variable HOSTNAME {
                val [ the-fw ];
            }
        }
    }
    scaling {
        min_active 1;
        max_active 1;
    }
    extensions {
        extension NSO {
            properties {
                property VDU {
                    value firewall;
                }
                property VNFR {
                    value ASA;
                }
            }
        }
    }
}
```

A network named *volvo-fw-router-inside-net* has been created, this network is used to stitch the CSR to the ASA.

Two VM groups are created, one for each VDU.

Importing (onboarding)

The simplest way of importing descriptors into NSO is to use the "load merge" functionality.

```
# load merge vnf-catalogue.xml
# commit
```

To export descriptors to file, the `| display` and `| save` are handy.

```
# show full-configuration nfvo vnfd CSR1kv | display xml | save csr-vnfd.xml
```

or

```
# show full-configuration nfvo vnfd CSR1kv | display json | save csr-vnfd.json
```

Onboarding flavor and image to the VIM

To create a flavor for an ASA on esc0:

```
# set nfvo onboarding esc flavor ASAFlavor vnfd ASA vdu firewall esc-device esc0
# commit
```

This will result in the following configuration being set in ESC:

```
# show full-configuration devices device esc0 config esc:esc_datamodel flavors flavor ASAFlavor
description  "Flavor generated by NSO from VNFD 'ASA' and VDU 'firewall'";
vcpus        2;
memory_mb    4096;
root_disk_mb 10240;
```

To create an image for the same ASA on esc0:

```
# set nfvo onboarding esc image ASAImage vnfd ASA vdu firewall esc-device esc0
# commit
```

And this results in the following in the ESC:

```
# show full-configuration devices device esc0 config esc:esc_datamodel images image ASAImage
src              http://10.147.46.245/nfvo-modelling/asav943.qcow2;
disk_format      qcow2;
container_format bare;
```

Catalogue Examples

Example 4. VLD Catalogue

```
{
  "data": {
    "tailf-nfvo:nfvo": {
      "vld": [
        {
          "id": "ipv4-network"
        }
      ]
    }
  }
}
```

Example 5. VNFD Catalogue

```
{
  "data": {
    "tailf-nfvo:nfvo": {
      "vnfd": [
        {
          "id": "ASA",
          "name": "virtual ASA",
          "provider": "Cisco",
          "version": "9.4.3",
          "description": "Virtual security appliance",
          "vdu": [
            {
              "id": "firewall",
              "internal-connection-point": [
                {
                  "id": "inside",
                  "external-connection-point": "cp-inside",
                  "interface-id": 1
                },
                {
                  "id": "mgmt",
                  "external-connection-point": "cp-mgmt",
                  "interface-id": 0
                },
                {
                  "id": "outside",
                  "external-connection-point": "cp-outside",
                  "interface-id": 2
                }
              ],
              "virtual-compute": {
                "virtual-memory": {
                  "virtual-memory-size": "4.0"
                },
                "virtual-cpu": {
```

```
            "number-of-virtual-cpus": 2
          }
        },
        "virtual-storage": [
          {
            "id": "root",
            "type-of-storage": "root",
            "size-of-storage": 10
          }
        ],
        "software-image": {
          "container-format": "bare",
          "disk-format": "qcow2",
          "image": "http://10.147.46.245/nfvo-modelling/asav943.qcow2"
        },
        "device-type": {
          "cli": {
            "ned-id": "asa-id:cisco-asa"
          }
        },
        "bootup-time": 300,
        "recovery-wait-time": 60
      }
    ],
    "external-connection-point": [
      {
        "id": "cp-inside"
      },
      {
        "id": "cp-mgmt",
        "management": [null]
      },
      {
        "id": "cp-outside"
      }
    ],
    "deployment-flavor": [
      {
        "id": "dep-fw",
        "vdu-profile": [
          {
            "vdu": "firewall",
            "min-number-of-instances": 1,
            "max-number-of-instances": 1
          }
        ],
        "instantiation-level": [
          {
            "id": "small",
            "vdu-level": [
              {
                "vdu": "firewall",
                "number-of-instances": 1
              }
            ]
          }
        ]
      }
    ]
  },
  {
    "id": "CSR1kv",
```

```
"name": "CSR 1000v",
"provider": "Cisco",
"version": "1.0",
"description": "Cloud router",
"vdu": [
  {
    "id": "CSR",
    "internal-connection-point": [
      {
        "id": "left",
        "external-connection-point": "left",
        "interface-id": 1
      },
      {
        "id": "mgmt",
        "external-connection-point": "mgmt",
        "interface-id": 0
      },
      {
        "id": "right",
        "external-connection-point": "right",
        "interface-id": 2
      }
    ],
    "virtual-compute": {
      "virtual-memory": {
        "virtual-memory-size": "3.0"
      },
      "virtual-cpu": {
        "number-of-virtual-cpus": 1
      }
    },
    "virtual-storage": [
      {
        "id": "root",
        "type-of-storage": "root",
        "size-of-storage": 8
      }
    ],
    "software-image": {
      "container-format": "bare",
      "disk-format": "qcow2",
      "image": "http://10.147.46.245/nso-demo/csr1000v-universalk9.03.14.01.S.155-1
      "additional-setting": [
        {
          "id": "disk_bus",
          "value": "virtio"
        },
        {
          "id": "e1000_net",
          "value": "false"
        },
        {
          "id": "serial_console",
          "value": "true"
        },
        {
          "id": "virtio_net",
          "value": "false"
        }
      ]
    },
```

```
      "device-type": {
        "cli": {
          "ned-id": "ios-id:cisco-ios"
        }
      },
      "bootup-time": 300,
      "recovery-wait-time": 120,
      "day0": [
        {
          "destination": "iosxe_config.txt",
          "mandatory": [null]
        }
      ]
    }
  ],
  "external-connection-point": [
    {
      "id": "left"
    },
    {
      "id": "mgmt",
      "management": [null]
    },
    {
      "id": "right"
    }
  ],
  "deployment-flavor": [
    {
      "id": "basic",
      "vdu-profile": [
        {
          "vdu": "CSR",
          "min-number-of-instances": 1,
          "max-number-of-instances": 10
        }
      ],
      "instantiation-level": [
        {
          "id": "basic",
          "vdu-level": [
            {
              "vdu": "CSR",
              "number-of-instances": 1
            }
          ]
        },
        {
          "id": "gold",
          "vdu-level": [
            {
              "vdu": "CSR",
              "number-of-instances": 4
            }
          ]
        }
      ],
      "default-instantiation-level": "basic"
    }
  ]
},
{
```

```
"id": "QVPC-DI",
"name": "The QVPC",
"provider": "cisco",
"version": "0.1",
"description": "QVPC-DI",
"vdu": [
  {
    "id": "CF-1",
    "internal-connection-point": [
      {
        "id": "core",
        "virtual-link": "core",
        "interface-id": 1
      },
      {
        "id": "internal",
        "virtual-link": "di-internal",
        "interface-id": 2
      },
      {
        "id": "mgmt",
        "virtual-link": "mgmt",
        "interface-id": 0
      }
    ],
    "virtual-compute": {
      "virtual-memory": {
        "virtual-memory-size": "4.0"
      },
      "virtual-cpu": {
        "number-of-virtual-cpus": 2
      }
    },
    "software-image": {
      "name": "Image for CF-1",
      "container-format": "bare",
      "disk-format": "qcow2",
      "image": "http://foo/cf1.qcow2"
    },
    "bootup-time": 120,
    "recovery-wait-time": 120
  },
  {
    "id": "CF-2",
    "internal-connection-point": [
      {
        "id": "core",
        "virtual-link": "core",
        "interface-id": 1
      },
      {
        "id": "internal",
        "virtual-link": "di-internal",
        "interface-id": 2
      },
      {
        "id": "mgmt",
        "description": "",
        "virtual-link": "mgmt",
        "interface-id": 0
      }
    ],
```

```
      "virtual-compute": {
        "virtual-memory": {
          "virtual-memory-size": "4.0"
        },
        "virtual-cpu": {
          "number-of-virtual-cpus": 2
        }
      },
      "bootup-time": 120,
      "recovery-wait-time": 120
    },
    {
      "id": "SF",
      "internal-connection-point": [
        {
          "id": "core",
          "description": "",
          "virtual-link": "core",
          "interface-id": 1
        },
        {
          "id": "internal",
          "description": "",
          "virtual-link": "di-internal",
          "interface-id": 2
        },
        {
          "id": "service",
          "description": "",
          "external-connection-point": "service",
          "interface-id": 0
        }
      ],
      "virtual-compute": {
        "virtual-memory": {
          "virtual-memory-size": "4.0"
        },
        "virtual-cpu": {
          "number-of-virtual-cpus": 2
        }
      },
      "bootup-time": 120,
      "recovery-wait-time": 120
    }
  ],
  "virtual-link": [
    {
      "id": "core",
      "layer-protocol": "IPv4"
    },
    {
      "id": "di-internal",
      "layer-protocol": "IPv4"
    },
    {
      "id": "mgmt",
      "layer-protocol": "IPv4"
    }
  ],
  "external-connection-point": [
    {
      "id": "core",
```

```
      "virtual-link": "core"
    },
    {
      "id": "mgmt",
      "virtual-link": "mgmt"
    },
    {
      "id": "service"
    }
  ],
  "deployment-flavor": [
    {
      "id": "the-flavor",
      "vdu-profile": [
        {
          "vdu": "CF-1",
          "min-number-of-instances": 1,
          "max-number-of-instances": 10
        },
        {
          "vdu": "CF-2",
          "min-number-of-instances": 1,
          "max-number-of-instances": 10
        },
        {
          "vdu": "SF",
          "min-number-of-instances": 1,
          "max-number-of-instances": 40
        }
      ],
      "instantiation-level": [
        {
          "id": "gold",
          "vdu-level": [
            {
              "vdu": "CF-1",
              "number-of-instances": 10
            },
            {
              "vdu": "CF-2",
              "number-of-instances": 10
            },
            {
              "vdu": "SF",
              "number-of-instances": 20
            }
          ]
        },
        {
          "id": "silver",
          "vdu-level": [
            {
              "vdu": "CF-1",
              "number-of-instances": 1
            },
            {
              "vdu": "CF-2",
              "number-of-instances": 1
            },
            {
              "vdu": "SF",
              "number-of-instances": 1
```

```
            }
          ]
        }
      ],
      "default-instantiation-level": "silver"
    }
  ]
},
{
  "id": "two-machine-cirros",
  "name": "Tow machine Cirros",
  "provider": "Cisco",
  "version": "1.0",
  "description": "VNFD with two machines",
  "vdu": [
    {
      "id": "cirros",
      "description": "The Cirros Image",
      "internal-connection-point": [
        {
          "id": "internal",
          "virtual-link": "di-internal",
          "interface-id": 1
        },
        {
          "id": "mgmt",
          "external-connection-point": "ext-mgmt",
          "interface-id": 0
        }
      ],
      "virtual-compute": {
        "virtual-memory": {
          "virtual-memory-size": "0.5"
        },
        "virtual-cpu": {
          "number-of-virtual-cpus": 1
        }
      },
      "virtual-storage": [
        {
          "id": "root",
          "type-of-storage": "root",
          "size-of-storage": 1
        }
      ],
      "bootup-time": 120,
      "recovery-wait-time": 15
    }
  ],
  "virtual-link": [
    {
      "id": "di-internal",
      "description": "The internal network",
      "layer-protocol": "IPv4"
    }
  ],
  "external-connection-point": [
    {
      "id": "ext-mgmt",
      "management": [null]
    }
  ],
```

```
          "deployment-flavor": [
            {
              "id": "first-flavor",
              "description": "The only flavor here",
              "vdu-profile": [
                {
                  "vdu": "cirros",
                  "min-number-of-instances": 2,
                  "max-number-of-instances": 2
                }
              ],
              "instantiation-level": [
                {
                  "id": "il-1",
                  "vdu-level": [
                    {
                      "vdu": "cirros",
                      "number-of-instances": 2
                    }
                  ]
                }
              ],
              "default-instantiation-level": "il-1"
            }
          ]
        }
      ]
    }
  }
}
```

Example 6. NSD Catalogue

```
{
  "data": {
    "tailf-nfvo:nfvo": {
      "nsd": [
        {
          "id": "fw-router",
          "version": "1.0",
          "vnfd": [
            {
              "vnfd": "ASA",
              "connection-point": [
                {
                  "id": "cp-inside",
                  "virtual-link": "inside-net"
                },
                {
                  "id": "cp-mgmt",
                  "service-access-point": "mgmt"
                },
                {
                  "id": "cp-outside",
                  "service-access-point": "outside"
                }
              ]
            },
            {
              "vnfd": "CSR1kv",
              "connection-point": [
                {
                  "id": "left",
```

```
                "virtual-link": "inside-net"
              },
              {
                "id": "mgmt",
                "service-access-point": "mgmt"
              },
              {
                "id": "right",
                "service-access-point": "inside"
              }
            ]
          }
        ],
        "service-access-point": [
          {
            "id": "inside"
          },
          {
            "id": "mgmt"
          },
          {
            "id": "outside"
          }
        ],
        "virtual-link": [
          {
            "id": "inside-net",
            "layer-protocol": "IPv4"
          }
        ]
      }
    ]
  }
 }
}
```

Migrating from the VM Manager

If you have an existing VM Manager deployment, this section provides an example on how to migrate to the NFVO Bundle.

```
vm volvo_NewYorkrouter {
    username admin;
    esc {
        deployment-name volvo_NewYorkrouter;
        authgroup       csr;
        tenant          volvo;
        vm-device       esc0;
        service-name    CSR;
        service-version 1.0;
        vm-group        CSR;
        day0 {
            source-url       http://10.147.46.245/nso-demo/CSR_day0.txt;
            destination-file iosxe_config.txt;
            variables {
                variable NAME {
                    value volvo_volvo_NewYorkrouter_CSR_esc0;
                }
            }
        }
        interface 0 {
            name net_osmgmt;
```

```
            ip   192.168.131.129;
        }
        interface 1 {
            name p_net;
            ip   10.1.1.129;
        }
        interface 2 {
            name ce_net;
            ip   10.2.2.129;
        }
        scaling-min     1;
        scaling-max     1;
        scaling-pool ce_net {
            ip [ 10.2.2.129 ];
        }
        scaling-pool net_osmgmt {
            ip [ 192.168.131.129 ];
        }
        scaling-pool p_net {
            ip [ 10.1.1.129 ];
        }
        device-type {
            cli {
                ned-id   cisco-ios;
                protocol ssh;
            }
        }
    }
}
```

You'd first have to create a VNFD for this router. A minimal one could look as follows

```
# show full-configuration nfvo vnfd CSR1kv
name        "CSR 1000v";
provider    Cisco;
version     1.0;
description "Cloud router";
vdu CSR {
    internal-connection-point left {
        external-connection-point left;
        interface-id              1;
    }
    internal-connection-point mgmt {
        external-connection-point mgmt;
        interface-id              0;
    }
    internal-connection-point right {
        external-connection-point right;
        interface-id              2;
    }
    virtual-compute {
        virtual-memory {
            virtual-memory-size 4.0;
        }
        virtual-cpu {
            number-of-virtual-cpus 2;
        }
    }
    virtual-storage root {
        type-of-storage root;
        size-of-storage 8;
    }
    software-image {
```

```
        container-format bare;
        disk-format      qcow2;
        image            http://10.147.46.245/nso-demo/csr1000v-universalk9.03.14.01.S.155-1.S1-
        additional-setting disk_bus {
            value virtio;
        }
        additional-setting e1000_net {
            value false;
        }
        additional-setting serial_console {
            value true;
        }
        additional-setting virtio_net {
            value false;
        }
    }
    device-type {
        cli {
            ned-id ios-id:cisco-ios;
        }
    }
    bootup-time        300;
    recovery-wait-time 120;
    day0 iosxe_config.txt {
        mandatory;
    }
}
external-connection-point left;
external-connection-point mgmt {
    management;
}
external-connection-point right;
deployment-flavor basic {
    vdu-profile CSR {
        min-number-of-instances 1;
        max-number-of-instances 1;
    }
    instantiation-level basic {
        vdu-level CSR {
            number-of-instances 1;
        }
    }
    default-instantiation-level basic;
}
```

In the VNFD you will specify VM's connections points, please see the *external-connection-points internal-connection-points*. If you have the parameters for the image and flavor, please modify the *virtual-compute*, *software-image* and *virtual-storage* accordingly. If you don't have them, you will have to reference existing OpenStack flavors and images when instantiating the VNF. Please also note the device type (Cisco IOS NED).

The *deployment-flavor* and *instantiation-level* dictates the scaling parameters, in this case only one CSR will be created.

To start a VM, you'll have to create a VNFR, e.g.

```
# show full-configuration nfvo vnfr esc vnf-deployment volvo volvo_NewYorkrouter
username admin;
esc-device esc0;
vnfr CSR1kvNTT {
    vnfd-flavor         basic;
    instantiation-level basic;
```

```
    managed;
    vdu CSR {
        image-name  csr;
        flavor-name csr;
        day0 iosxe_config.txt {
            url http://10.147.46.245/nso-demo/CSR_day0.txt;
            variable NAME {
               value volvo_volvo_NewYorkrouter_CSR_esc0;
            }
        }
        authgroup   csr;
        connection-point-address mgmt {
            start 192.168.131.129;
            end   192.168.131.129;
        }
        connection-point-address left {
            start 10.1.1.129;
            end   10.1.1.129;
        }
        connection-point-address right {
            start 10.2.2.129;
            end   10.2.2.129;
        }
    }
    vnfd-connection-point mgmt {
        vlr net_osmgmt;
    }
    vnfd-connection-point left {
        vlr p_net;
    }
    vnfd-connection-point right {
        vlr ce_net;
    }
}
```

Worth noticing here is the specification of day0 URL and variables. You also reference OpenStack flavors (csr) and images (csr) here.

The managed flag indicates this VNF should be added to NSO's device tree when booted.

To specify the address on a network interface, e.g. mgmt, you specify the address range (just one address in this case) on the VDU's connection point. The mapping to an OpenStack network is done through a reference to a VLR with the correct name, e.g. net_osmgmt.

```
# show full-configuration nfvo vlr esc vlr
vlr net_osmgmt;
vlr ce_net;
vlr p_net;
```

ESC Tips and Tricks

This sections contains some tips and tricks that may be helpful when developing on the ESC.

Logging in to the ESC

You can SSH into ESC using username *admin* and password *cisco123*.

Log files

Sometimes it is helpful to scan through the ESC's logfiles. Usually, the cause of errors can be found in `/var/log/esc/escmanager.log`.

ESC ConfD CLI

It is possible to start the ConfD CLI on ESC to look at the data models.

```
# /opt/cisco/esc/confd/bin/confd_cli -u admin -C
# show running-config
...
esc_datamodel tenants tenant CustomerA
 deployments deployment CustomerA_CleanPipesrouter
  vm_group CSR
   bootup_time        600
   recovery_wait_time 0
   interfaces interface 0
    network    net_osmgmt
    ip_address 192.168.131.129
   !
   interfaces interface 1
    network    p_net
    ip_address 10.1.1.129
   !
   interfaces interface 2
    network    ce_net
    ip_address 10.2.2.129
   !
   kpi_data kpi VM_ALIVE
    metric_value             60
    metric_cond              GT
    metric_type              UINT32
    metric_occurrences_true  3
    metric_occurrences_false 3
    metric_collector type ICMPPing
    metric_collector nicid 0
    metric_collector poll_frequency 15
    metric_collector polling_unit seconds
...
```

ESC Web UI

ESC's web UI is available on http://ESC-IP:9000.

You can SSH into ESC using username *admin* and password *cisco123*.

In here you can find deployments, logs, version information, OpenStack hypervisor information, etc.

CHAPTER 4

Developing for the NFVO Bundle

Introduction

This chapter contains information on how to develop services on top of the bundle.

Example Services

Instantiating a VNFD in Java and Python

This is a simple service that instantiates a CSR1000v with some simple input parameters. Please see the CSR-service service in Example 9, “Example YANG Models”.

Input parameters

tenant	Name of the tenant
deployment-name	Deployment name
authgroup	Authgroup used when connecting to the device
csr-hostname	This name will be sent through day0 to the device
left-net	Network to connect the 'left' interface to
mgmt-net	Network to connect the 'mgmt' interface to
right-net	Network to connect the 'right' interface to

Example 7. Service code in Python

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service, PlanComponent
import ncs.maapi as maapi
import ncs.maagic as maagic
from common import authgroup_password, setup_crypto


class CSRService(Service):
    @Service.create
```

```
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')
        self.self_plan = PlanComponent(service, 'self', 'ncs:self')
        self.self_plan.append_state('ncs:init')
        self.self_plan.append_state('ncs:ready')
        self.self_plan.set_reached('ncs:init')

        setup_crypto(tctx)
        pwd = authgroup_password(tctx, root, service.authgroup)

        vars = ncs.template.Variables()
        vars.add('ADMIN_PWD', pwd)
        vars.add('USERNAME', tctx.username)
        self.template = ncs.template.Template(service)
        self.template.apply('tailf-nfvo-examples-vnfr', vars)
        self.template.apply('tailf-nfvo-examples-kicker-py')

        with maapi.single_read_trans(tctx.username, "system",
                                     db=ncs.OPERATIONAL) as oper_th:
            oroot = maagic.get_root(oper_th)
            vnf_deployment = oroot.nfvo__nfvo.vnfr.esc.vnf_deployment
            if (service.tenant, service.deployment_name) not in vnf_deployment:
                self.log.info("VNFR not yet ready - service not ready")
                return

            dep = vnf_deployment[service.tenant, service.deployment_name]
            ready_status = str(dep.plan.component['self'].state['ready'].status)
            if ready_status != 'reached':
                self.log.info("VNFR not yet ready - not reached")
                return

            # Get the created device
            self.log.info('Getting device')
            res = root.nfvo__nfvo.vnfr.esc.vnf_deployment_result[service.tenant,
                                                                 service.deployment_name]
            csr_name = next(iter(res.vdu['CSR1kv', 'CSR'].vm_device)).device_name
            self.log.info("CSR name %s" % (csr_name))

            vars = ncs.template.Variables()
            vars.add('DEVICE_NAME', csr_name)
            self.template.apply('tailf-nfvo-examples-snmp-server', vars)
            self.self_plan.set_reached('ncs:ready')
```

Example 8. Service code in Java

```
package com.tailf.nfvo.examples;

import com.tailf.conf.*;
import com.tailf.dp.annotations.ServiceCallback;
import com.tailf.dp.proto.ServiceCBType;
import com.tailf.dp.services.ServiceContext;
import com.tailf.maapi.Maapi;
import com.tailf.maapi.MaapiCrypto;
import com.tailf.navu.*;
import com.tailf.ncs.PlanComponent;
import com.tailf.ncs.ns.Ncs;
import com.tailf.ncs.template.Template;
import com.tailf.ncs.template.TemplateVariables;
import com.tailf.nfvo.examples.namespaces.tailfNfvoExamples;
import org.apache.log4j.Logger;

import java.io.IOException;
import java.util.Properties;
```

```
public class CSR {
    private final Logger LOGGER = Logger.getLogger(CSR.class);

    @ServiceCallback(servicePoint="tailf-nfvo-examples-csr-service-java",
        callType= ServiceCBType.CREATE)
    public Properties create(ServiceContext context,
                             NavuNode service,
                             NavuNode ncsRoot,
                             Properties opaque)
        throws ConfException, IOException {
        PlanComponent selfPlan = new PlanComponent(service, "self", "ncs:self");
        selfPlan.appendState("ncs:init");
        selfPlan.appendState("ncs:ready");
        selfPlan.setReached("ncs:init");

        Maapi m = ncsRoot.context().getMaapi();
        int userSessId = m.getUserSessions()[0];
        String username = m.getUserSession(userSessId).getUser();

        String pwd = authgroupPassword(ncsRoot, username, service.leaf(tailfNfvoExamples._aut

        TemplateVariables vars = new TemplateVariables();
        vars.putQuoted("ADMIN_PWD", pwd);
        vars.putQuoted("USERNAME", username);

        Template vnfr = new Template(context, "tailf-nfvo-examples-vnfr");
        vnfr.apply(service, vars);
        Template alloc = new Template(context, "tailf-nfvo-examples-kicker-java");
        alloc.apply(service, null);


        NavuContext ctx = new NavuContext(m);

        ctx.startOperationalTrans(Conf.MODE_READ);

        try {
            NavuNode opRoot = new NavuContainer(ctx);

            final String tenant = service.leaf(tailfNfvoExamples._tenant_).valueAsString();
            final String depName = service.leaf(tailfNfvoExamples._deployment_name_).valueAsS
            String readyPath = String.format("/nfvo:nfvo/vnfr/nfvo-esc:esc/vnf-deployment{%s
                    tenant,
                    depName);

            NavuNode readyState;
            try {
                readyState = opRoot.getNavuNode(new ConfPath(readyPath));
            }
            catch (NavuException e) {
                LOGGER.info("Plan not yet available");
                return opaque;
            }

            if (!"reached".equals(readyState.leaf(Ncs._status_).valueAsString())) {
                LOGGER.info("Plan not yet ready");
            }

            // Get the CSR device
            NavuList vmDevices = (NavuList) opRoot.getNavuNode(new ConfPath("/nfvo:nfvo/vnfr/
                    "/vdu{CSR1kv CSR}/vm-device",
                    tenant, depName));
```

```
            String deviceName = vmDevices.elements().iterator().next().leaf("device-name").value

            LOGGER.info("CSR device name " + deviceName);

            TemplateVariables deviceVars = new TemplateVariables();
            deviceVars.putQuoted("DEVICE_NAME", deviceName);
            Template deviceTemplate = new Template(context, "tailf-nfvo-examples-snmp-server");
            deviceTemplate.apply(service, deviceVars);
            selfPlan.setReached("ncs:ready");
        }
        finally {
            ctx.finishClearTrans();
        }

        return opaque;
    }

    private String authgroupPassword(NavuNode ncsRoot, String username, String authgrp) throws C
        NavuContainer ag = ncsRoot.container(Ncs._devices_).container(Ncs._authgroups_).list(Ncs

        String encPwd;
        if (ag.list(Ncs._umap_).containsNode(username)) {
            encPwd = ag.list(Ncs._umap_).elem(username).leaf(Ncs._remote_password_).valueAsStrin
        }
        else {
            encPwd = ag.container(Ncs._default_map_).leaf(Ncs._remote_password_).valueAsString()
        }

        MaapiCrypto mc = new MaapiCrypto(ncsRoot.context().getMaapi());
        return mc.decrypt(encPwd);
    }

}
```

First the service creates its plan and marks the ready state as *init*. Then it decrypts the password for the provided authentication group. This clear text password is used for day0.

Then the service applies its vnfr template, shown below, to instantiate the VNF.

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <nfvo xmlns="http://tail-f.com/pkg/nfvo">
    <vnfr>
      <esc xmlns="http://tail-f.com/pkg/tailf-nfvo-esc">
        <vnf-deployment>
          <tenant>{/tenant}</tenant>
          <deployment-name>{/deployment-name}</deployment-name>
          <username>{$USERNAME}</username>
          <esc-device>
            <name>esc0</name>
          </esc-device>
          <vnfr>
            <id>CSR1kv</id>
            <vnfd-flavor>basic</vnfd-flavor>
            <instantiation-level>basic</instantiation-level>
            <managed/>
            <vdu>
              <id>CSR</id>
              <image-name>CSRImage</image-name>
              <flavor-name>CSRFlavor</flavor-name>
              <day0>
                <destination>iosxe_config.txt</destination>
                <url>http://10.147.46.245/nfvo-modelling/CSR_day0.txt</url>
```

```
                <variable>
                  <name>ADMIN_PWD</name>
                  <value>{$ADMIN_PWD}</value>
                </variable>
                <variable>
                  <name>HOSTNAME</name>
                  <value>{/csr-hostname}</value>
                </variable>
              </day0>
              <authgroup>{/authgroup}</authgroup>
            </vdu>
            <vnfd-connection-point>
              <id>left</id>
              <vlr>internet</vlr>
            </vnfd-connection-point>
            <vnfd-connection-point>
              <id>mgmt</id>
              <vlr>p_net</vlr>
            </vnfd-connection-point>
            <vnfd-connection-point>
              <id>right</id>
              <vlr>volvo-internal</vlr>
            </vnfd-connection-point>
          </vnfr>
        </vnf-deployment>
      </esc>
    </vnfr>
  </nfvo>
</config>
```

It then checks if the VNFR is ready and When it becomes ready, the SNMP server configuration is applied to the VNF device.

Instantiating a NSD in Python

This is a service that instantiates a CSR1kv and an ASAv, using a NS, with some simple input parameters. Please see the fw-router-service in Example 9, “Example YANG Models”.

Input parameters

tenant	Name of the tenant
deployment-name	Deployment name
csr-authgroup	Authgroup used when connecting to the CSR device
asa-authgroup	Authgroup used when connecting to the ASA device
csr-hostname	This name will be sent through day0 to the CSR device
asa-hostname	This name will be sent through day0 to the ASA device
left-net	Network to connect the 'left' interface to
mgmt-net	Network to connect the 'mgmt' interface to
right-net	Network to connect the 'right' interface to

```
# -*- mode: python; python-indent: 4 -*-
import ncs
from ncs.application import Service, PlanComponent
```

```
from common import authgroup_password, setup_crypto


class FWRService(Service):
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('Service create(service=', service._path, ')')
        self.template = ncs.template.Template(service)

        self.self_plan = PlanComponent(service, 'self', 'ncs:self')
        self.self_plan.append_state('ncs:init')
        self.self_plan.append_state('ncs:ready')
        self.self_plan.set_reached('ncs:init')

        self.nsr_plan = PlanComponent(service, 'nsr', 'tailf-nfvo-examples:nsr')
        self.nsr_plan.append_state('ncs:init')
        self.nsr_plan.append_state('ncs:ready')

        vars = ncs.template.Variables()
        vars.add('USERNAME', tctx.username)

        setup_crypto(tctx)

        csr_pwd = authgroup_password(tctx, root, service.csr_authgroup)
        vars.add('CSR_PWD', csr_pwd)
        asa_pwd = authgroup_password(tctx, root, service.asa_authgroup)
        vars.add('ASA_PWD', asa_pwd)

        self.template.apply('tailf-nfvo-examples-nsr', vars)
        self.nsr_plan.set_reached('ncs:init')

        with ncs.maapi.single_read_trans(tctx.username, tctx.context,
                                         db=ncs.OPERATIONAL) as oper_th:
            oroot = ncs.maagic.get_root(oper_th)
            nsr_list = oroot.nfvo__nfvo.nsr.esc.nsr
            if not (service.tenant, service.deployment_name) in nsr_list:
                self.log.info("NSR not yet ready")
                return
            nsr = nsr_list[service.tenant, service.deployment_name]
            if 'self' not in nsr.plan.component:
                self.log.info("NSR not yet ready")
                return

            status = nsr.plan.component['self'].state['ready'].status

            if 'reached' != str(status):
                self.log.info("NSR not yet ready")
                return

            # Get the devices from the VNFR
            dep = oroot.nfvo__nfvo.vnfr
            vnf_deployment = oroot.nfvo__nfvo.vnfr.esc.vnf_deployment_result
            dep = vnf_deployment[service.tenant, service.deployment_name]

            device_name = next(iter(dep.vdu['CSR1kv', 'CSR'].vm_device)).device_name
            self.log.info("CSR name %s" % (device_name))

            vars = ncs.template.Variables()
            vars.add('DEVICE_NAME', device_name)
            self.template.apply('tailf-nfvo-examples-snmp-server', vars)

            device_name = next(iter(dep.vdu['ASA', 'firewall'].vm_device)).device_name
```

```
            self.log.info("ASA name %s" % (device_name))

            vars = ncs.template.Variables()
            vars.add('DEVICE_NAME', device_name)
            self.template.apply('tailf-nfvo-examples-snmp-server', vars)

        self.nsr_plan.set_reached('ncs:ready')
        self.self_plan.set_reached('ncs:ready')
```

The service code works the same way as the CSR service, please see the section called “Instantiating a VNFD in Java and Python”.

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <nfvo xmlns="http://tail-f.com/pkg/nfvo">
    <nsr>
      <esc xmlns="http://tail-f.com/pkg/tailf-nfvo-esc">
        <nsr>
          <tenant>{/tenant}</tenant>
          <deployment-name>{/deployment-name}</deployment-name>
          <username>{$USERNAME}</username>
          <nsd>fw-router</nsd>
          <esc-device>
            <name>esc0</name>
          </esc-device>
          <vnfr>
            <id>ASA</id>
            <vnfd-flavor>dep-fw</vnfd-flavor>
            <instantiation-level>small</instantiation-level>
            <managed/>
            <vdu>
              <id>firewall</id>
              <image-name>ASAImage</image-name>
              <flavor-name>ASAFlavor</flavor-name>
              <day0>
                <destination>day0-config</destination>
                <url>http://10.147.46.245/nfvo-modelling/ASA_941_day0.txt</url>
                <variable>
                  <name>ADMIN_PWD</name>
                  <value>{$ASA_PWD}</value>
                </variable>
                <variable>
                  <name>HOSTNAME</name>
                  <value>{/asa-hostname}</value>
                </variable>
              </day0>
              <authgroup>{/asa-authgroup}</authgroup>
            </vdu>
          </vnfr>
          <vnfr>
            <id>CSR1kv</id>
            <vnfd-flavor>basic</vnfd-flavor>
            <instantiation-level>basic</instantiation-level>
            <managed/>
            <vdu>
              <id>CSR</id>
              <image-name>CSRImage</image-name>
              <flavor-name>CSRFlavor</flavor-name>
              <day0>
                <destination>iosxe_config.txt</destination>
                <url>http://10.147.46.245/nfvo-modelling/CSR_day0.txt</url>
                <variable>
                  <name>ADMIN_PWD</name>
```

```
                <value>{$CSR_PWD}</value>
              </variable>
              <variable>
                <name>HOSTNAME</name>
                <value>{/csr-hostname}</value>
              </variable>
            </day0>
            <authgroup>{/csr-authgroup}</authgroup>
          </vdu>
        </vnfr>
        <virtual-link>
          <id>inside-net</id>
          <dhcp/>
          <subnet>
            <network>10.0.0.0/24</network>
            <gateway>10.0.0.1</gateway>
          </subnet>
        </virtual-link>
        <service-access-point>
          <id>inside</id>
          <vlr>{/inside-net}</vlr>
        </service-access-point>
        <service-access-point>
          <id>mgmt</id>
          <vlr>{/mgmt-net}</vlr>
        </service-access-point>
        <service-access-point>
          <id>outside</id>
          <vlr>{/outside-net}</vlr>
        </service-access-point>
      </nsr>
    </esc>
  </nsr>
 </nfvo>
 <kickers xmlns="http://tail-f.com/ns/kicker">
   <data-kicker>
     <id>tailf-nfvo-example-fw-router-{/tenant}-{/deployment-name}</id>
     <monitor>/nfvo/nsr/nfvo-esc:esc/nsr[tenant='{/tenant}'][deployment-name='{/deployment-name
     <trigger-expr>nfvo-esc:status='reached'</trigger-expr>
     <kick-node xmlns:example="http://tail-f.com/pkg/nfvo-examples">
       /tailf-nfvo-examples/fw-router-service[tenant='{/tenant}'][deployment-name='{/deployment
     </kick-node>
     <action-name>reactive-re-deploy</action-name>
   </data-kicker>
 </kickers>
</config>
```

It then checks if the NSR is ready. When it becomes ready, the SNMP server configuration is applied to the VNFs.

Service Helper Code

The common module contains some helper functions for the services.

```
import ncs
from _ncs import decrypt


def setup_crypto(tctx):
    with ncs.maapi.Maapi() as m:
        m.attach(tctx)
        try:
```

```
            m.install_crypto_keys()
        finally:
            m.detach(tctx)


def authgroup_password(tctx, root, authgrp_name):
    authgrp = root.ncs__devices.authgroups.group[authgrp_name]
    username = tctx.username

    if username in authgrp.umap:
        encpwd = authgrp.umap[username].remote_password
    else:
        encpwd = authgrp.default_map.remote_password

    return decrypt(str(encpwd))
```

Service Models

Example 9. Example YANG Models

```
module tailf-nfvo-examples {

  namespace "http://tail-f.com/pkg/nfvo-examples";
  prefix tailf-nfvo-examples;

  import ietf-inet-types { prefix inet; }
  import tailf-common { prefix tailf; }
  import tailf-ncs { prefix ncs; }
  import tailf-nfvo { prefix nfvo; }
  import tailf-nfvo-esc { prefix nfvoe; }


  description
    "Example code for NFVO";

  revision 2016-01-01 {
    description
      "Initial revision.";
  }

  identity nsr {
    base ncs:plan-component-type;
  }

  grouping CSR-svc {
    leaf tenant {
      type string;
    }

    leaf deployment-name {
      type string;
    }

    leaf authgroup {
      mandatory true;
      type leafref {
        path "/ncs:devices/ncs:authgroups/ncs:group/ncs:name";
      }
    }

    leaf csr-hostname {
      mandatory true;
```

```
      type string;
    }

    leaf left-net {
      mandatory true;
      type leafref {
        path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
      }
    }
    leaf mgmt-net {
      mandatory true;
      type leafref {
        path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
      }
    }
    leaf right-net {
      mandatory true;
      type leafref {
        path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
      }
    }
  }

  container tailf-nfvo-examples {
    list CSR-service-python {
      uses ncs:plan-data;
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-examples-csr-service-python;
      key "tenant deployment-name";
      uses CSR-svc;
    } // CSR-service

    list CSR-service-java {
      uses ncs:plan-data;
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-examples-csr-service-java;
      key "tenant deployment-name";
      uses CSR-svc;
    }

    list fw-router-service {
      uses ncs:plan-data;
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-examples-fwr-service;
      key "tenant deployment-name";

      leaf tenant {
        type string;
      }

      leaf deployment-name {
        type string;
      }

      leaf csr-authgroup {
        mandatory true;
        type leafref {
          path "/ncs:devices/ncs:authgroups/ncs:group/ncs:name";
        }
      }

      leaf asa-authgroup {
```

```
        mandatory true;
        type leafref {
          path "/ncs:devices/ncs:authgroups/ncs:group/ncs:name";
        }
      }

      leaf csr-hostname {
        mandatory true;
        type string;
      }

      leaf asa-hostname {
        mandatory true;
        type string;
      }

      leaf outside-net {
        mandatory true;
        type leafref {
          path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
        }
      }
      leaf mgmt-net {
        mandatory true;
        type leafref {
          path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
        }
      }
      leaf inside-net {
        mandatory true;
        type leafref {
          path "/nfvo:nfvo/nfvo:vlr/nfvoe:esc/nfvoe:vlr/nfvoe:id";
        }
      }
    } // fw-router-service
  }
}
```

Staged Delete

If the deletion process of VNFs needs to be controlled, a staged delete can be used.

In some cases it is necessary to control the deletion process of virtual machines. In the default case all machines are stopped and deleted at the same time as the northbound service is removed. Some VNF has licenses enabled, at times it is necessary to return these licenses before the virtual machine is actually stopped. Staged delete allows a service developer to pause the deletion process before the VM is stopped. When the VM is ready to be deleted, the staged delete code allows the process to continue.

To enable staged delete, you need to do two things.

1. Set the staged-delete flag in the VNFR, see Example 17, “NFVO ESC YANG Models”.
2. Implement a staged delete handler, please see below.

A stage delete handler basically listens for changes to the staged delete data, please see Example 18, “Staged Delete Model” for the data models.

If a deployment is marked for staged delete, the service is not immediately deleted, but a delete request is written to the *delete* list. When the handler has processed the request, it is responsible for deleting the *delete* entry and write a *deleted* entry. The NFVO code will then delete devices and the actual service.

Note that we're using the experimental subscriber feature from the NSO 4.2 Python API.

```
# -*- mode: python; python-indent: 4 -*-
import ncs
import ncs.experimental
import ncs.maapi as maapi
import ncs.maagic as maagic


# ------------------------------------------------
# SUBSCRIBER ITERATOR OBJECT
# ------------------------------------------------
class StagedDeleteSubscriber(ncs.experimental.Subscriber):
    def init(self):
        # We're interested in changed to the 'delete' list
        self.register('/nfvo:nfvo/nfvo-esc-sd:staged-delete-esc/delete')

    def pre_iterate(self):
        return []

    def iterate(self, kp, op, oldv, newv, state):
        # Only care about entries added to the 'delete' list
        if op == ncs.MOP_CREATED:
            # Read tenant and deployment name
            key = (str(kp[0][0]), str(kp[0][1]))
            state.append(key)

        return ncs.ITER_CONTINUE

    # This will run in a separate thread
    def post_iterate(self, state):
        self.log.info('Revoker: RUNNING , state=' + str(state))

        username = ""
        context = "system"

        with maapi.single_write_trans("", context) as th:
            delete_l = maagic.get_node(th, '/nfvo:nfvo/nfvo-esc-sd:staged-delete-esc/delete')

            for k in state:
                #
                # At this point it is possible to react to the delete event
                # e.g. to return licenses for the devices k contains the key
                # for the deployment.
                # With this key it's possible to access
                # /nfvo/vnfr/esc/vnf-deployment-result to get the device list.

                username = str(delete_l[k].username)

                # Remove the entry from the delete list
                del delete_l[k]

                with maapi.single_write_trans(username, context) as uth:
                    deleted_l = maagic.get_node(uth,
                                                '/nfvo:nfvo/nfvo-esc-sd:staged-delete-esc/delete
                    # Create the entry in the deleted list
                    deleted = deleted_l.create(k)
                    deleted.username = username
                    self.log.info('Deleted:' + str(k))

                    # When we have written the deleted list entry, the
                    # devices and service will be deleted.
```

```
                    uth.apply()
                th.apply()
            self.log.info('Revoker: TERMINATING')

        # determine if post_iterate() should run
        def should_post_iterate(self, state):
            return state != []


# ---------------------------------------------
# COMPONENT THREAD THAT WILL BE STARTED BY NCS.
# ---------------------------------------------
class StagedDeleteHandler(ncs.application.Application):
    def setup(self):
        # The application class sets up logging for us.
        # It's a normal logging.getLogger() object.
        self.log.info('StagedDeleteHandler STARTED')

        self.sub = StagedDeleteSubscriber(self)
        self.sub.start()

        self.add_running_thread('StagedDeleteSubscriber')

    def teardown(self):
        # When the application is finished (which would happen if NCS went
        # down, packages were reloaded or some error occurred) this teardown
        # method will be called.

        # Inform the 'subscriber' that it has to shutdown
        self.sub.stop()

        self.del_running_thread('StagedDeleteSubscriber')
        self.log.info('StagedDeleteHandler FINISHED')
```

CHAPTER 5

The NFVO GUI

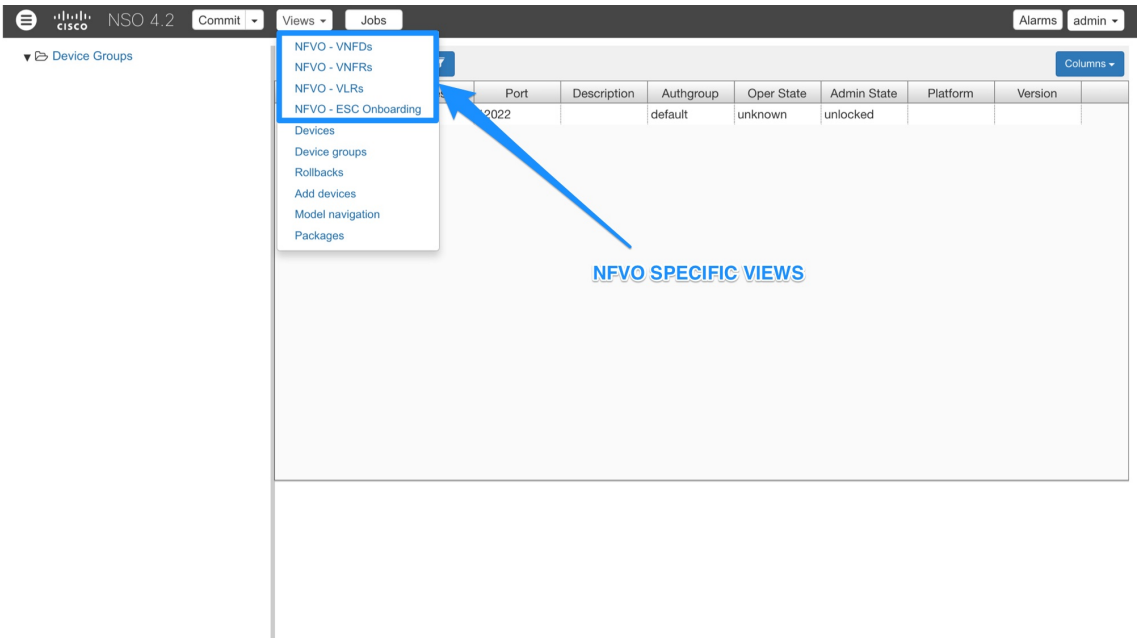
Introduction

blah TODO the new VIEWS

The NFVO Views

Blah bal balh

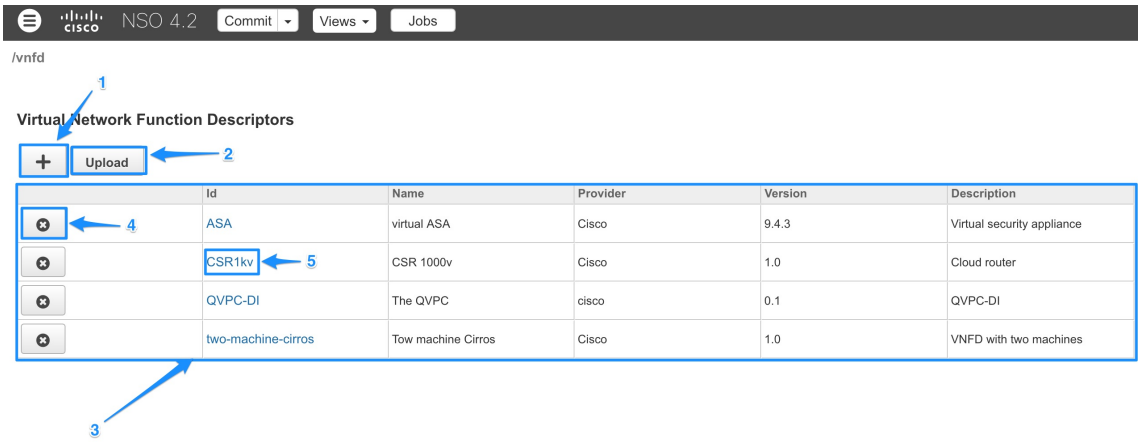
Figure 10. The NFVO Views



The VNFDs View

Blah bal balh

Figure 11. The VNFDs View

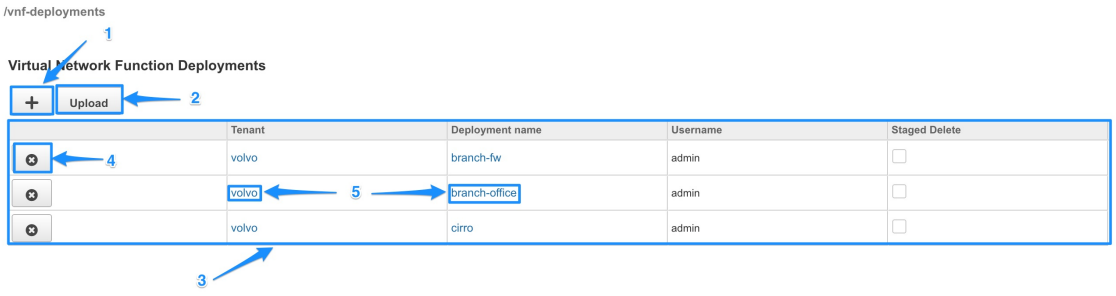


1. Create new VNFD
2. Upload config file containing one or several VNFD(s). Blah blah blah slbah Blah bal balh
3. List of VNFD(s)
4. Delete an VNFD instance
5. Link to a specific instance

The VNFRs View

Blah bal balh

Figure 12. The VNFRs View

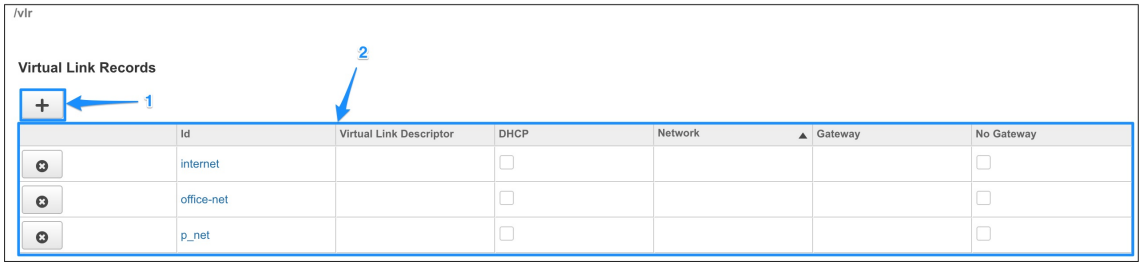


1. Create new VNFR deployment
2. Upload config file containing one or several VNFR Deployment(s)
3. List of VNFR Deployment(s)
4. Delete an VNFR instance
5. Link to a specific instance

The VLRs View

Blah bal balh

Figure 13. The VLRs View

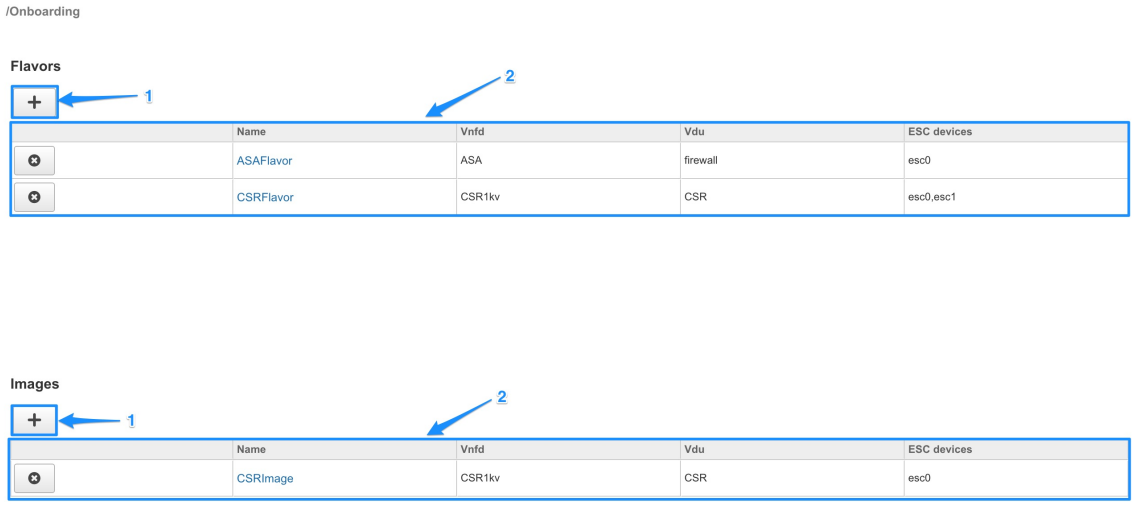


1 Create new VLR
2 List of VLR(s)

The ESC Onboarding View

Blah bal balh

Figure 14. The ESC Onboarding View



CHAPTER 6

The NFVO Bundle Data Models

Model Overview

Example 15. VNF YANG Models

```
submodule tailf-nfvo-vnf {
  belongs-to tailf-nfvo {
    prefix nfvo;
  }

  import tailf-common {
    prefix tailf;
  }
  import ietf-inet-types {
    prefix inet;
  }
  include tailf-nfvo-common {
    revision-date 2016-03-11;
  }

  description
    "Models for VNFD according to GS NFV IFA011.";

  revision 2016-03-24 {
    description
      "Initial revision.";
    reference "GS NFV IFA011";
  }

  grouping vnfd {
    list vnfd {
      key "id";
      description
        "A VNF Descriptor (VNFD) is a deployment template which describes a VNF
         in terms of deployment and operational behaviour requirements.
         It also contains connectivity, interface and virtualised resource
         requirements";
      reference "GS NFV IFA011: VNFD information element";
      leaf id {
        tailf:info "Unique ID of the VNFD";
        type string;
      }
```

```
leaf name {
  tailf:info "Human readable name of the VNFD";
  type string;
}
leaf provider {
  tailf:info "Provider of the VNF and of the VNFD";
  type string;
}
leaf version {
  tailf:info "Identifies the version of the VNFD";
  type string;
}
leaf description {
  tailf:info "Human readable description of the VNFD";
  type string;
}
list vdu {
  key "id";
  min-elements 1;
  description
    "The Virtualisation Deployment Unit (VDU) is a construct supporting
     the description of the deployment and operational behaviour of a
     VNF component, or the entire VNF if it was not componentized in
     components";
  reference "GS NFV IFA011: VDU information element";
  leaf id {
    tailf:info "Unique ID of the VDU";
    type string;
  }
  leaf description {
    tailf:info "Human readable description of the VDU";
    type string;
  }
  list internal-connection-point {
    key "id";
    unique "interface-id";
    min-elements 1;
    description
      "A internal-connection-point element is a type of connection point and describes
       network connectivity between a VDU instance and an internal Virtual Link or an
       external connection point.";
    reference "GS NFV IFA011: VduCpd information element";
    leaf id {
      type string;
    }
    leaf description {
      tailf:info "Human readable description of the connection point";
      type string;
    }
    choice cp-connection {
      mandatory true;
      description
        "A connection point must either connect to an internal virtual
         link or to an external connection points.";
      leaf virtual-link {
        tailf:info "Connect to an internal VLD";
        type leafref {
          path "../../../virtual-link/id";
        }
      }
      leaf external-connection-point {
        tailf:info "Connect to an external connection point";
```

```
      type leafref {
        path "../../../external-connection-point/id";
      }
    }
  }
  leaf interface-id {
    type uint8;
    mandatory true;
    description
      "Interface ID. This determines the order
       in which interfaces are presented to the guest VM";
  }
}
container virtual-compute {
  description
    "Specifies the sizing of the virtual machine.";
  container virtual-memory {
    leaf virtual-memory-size {
      tailf:info "Size in Gib";
      type decimal64 {
        fraction-digits 1;
        range "0..max";
      }
      units "GiB";
      mandatory true;
    }
  }
  container virtual-cpu {
    leaf number-of-virtual-cpus {
      type uint16 {
        range "1..max";
      }
      mandatory true;
    }
  }
}
list virtual-storage {
  key "id";
  description
    "Storage requirements for a Virtual Storage instance
     attached to the VNFC created from this VDU";
  leaf id {
    type string;
  }
  leaf type-of-storage {
    type enumeration {
      enum "root";
      enum "swap";
      enum "ephemeral";
    }
    mandatory true;
  }
  leaf size-of-storage {
    tailf:info "Size in Gib";
    type uint64;
    units "GiB";
    mandatory true;
  }
}
container software-image {
  presence true;
  description
```

```
      "VM image data for the image used to boot the VDU.";
    reference "GS NFV IFA011: SwImageDescriptor information element";
    leaf name {
      type string;
    }
    leaf container-format {
      mandatory true;
      type enumeration {
        enum "bare";
      }
      description
        "VIM specific information for the container format";
    }
    leaf disk-format {
      mandatory true;
      type enumeration {
        enum "qcow2";
        enum "raw";
        enum "vmdk";
      }
      description
        "VIM specific information for format of the image";
    }
    leaf image {
      mandatory true;
      type inet:uri;
      description
        "URI to the image";
    }
    list additional-setting {
      key "id";
      description
        "Can be used to pass additional information about
         this storage descriptior to the VNFM/VIM";
      leaf id {
        type string;
      }
      leaf value {
        type string;
      }
    }
  }
  container device-type {
    description
      "NED information needed to manage the device.";
    tailf:info "Management protocol for the device";
    choice ne-type {
      case netconf {
        leaf netconf {
          tailf:info "Use NETCONF to talk to the device";
          type empty;
        }
      }
      case generic {
        container generic {
          tailf:info "Use a generic NED to talk to the device";
          presence "use a generic ned to communicate with the device";
          leaf ned-id {
            tailf:info "The NED Identity";
            type string;
            mandatory true;
          }
```

```
      }
    }
    case cli {
      container cli {
        tailf:info "Use CLI to communicate with the device";
        presence "use CLI to communicate with the device";
        leaf ned-id {
          tailf:info "The NED Identity";
          type string;
          mandatory true;
        }
        leaf protocol {
          tailf:info "The CLI protocol";
          type enumeration {
            enum "telnet";
            enum "ssh";
          }
          default "ssh";
        }
        leaf port {
          type inet:port-number;
          description
            "Port for the management interface on the device.  If this leaf is
             not configured, a default value based on the type of device will
             be used. For example, a NETCONF device uses port 830, a
             CLI device over SSH uses port 22, and a SNMP device uses port 161.";
        }
      }
    }
  }
}
list depends-on {
  must "vdu != ../id" {
    error-message "A VDU cannot depend on itself";
  }
  key "vdu";
  description
    "Used to specify boot order between VDUs";
  leaf vdu {
    type leafref {
      path "../../../vdu/id";
    }
  }
}
leaf bootup-time {
  tailf:info "Maximum bootup time in seconds";
  type uint64;
  mandatory true;
}
leaf recovery-wait-time {
  tailf:info "Recovery wait time in seconds";
  type uint64;
  mandatory true;
}
list day0 {
  key "destination";
  description
    "The author can list the different day0 files here and
     mark the necessary files as mandatory. If the destination
     is mandatory, its existense can be checked at instantiation
     time.";
  leaf destination {
```

```
        tailf:info "Destination filename";
        type string;
      }
      leaf mandatory {
        tailf:info "Mandatory day0 file. At instantiation, the user have to provide this fil
        type empty;
      }
    }
  }
  list virtual-link {
    key "id";
    description
      "Represents the type of network connectivity mandated by the VNF
       vendor between two or more Connection";
    reference "GS NFV IFA011: Information elements related to the Vld";
    leaf id {
      type string;
    }
    leaf description {
      tailf:info "Human readable description of the VLD";
      type string;
    }
    leaf layer-protocol {
      tailf:info "Netowork type";
      type layer-protocol;
      mandatory true;
    }
  }
  list external-connection-point {
    key "id";
    min-elements 1;
    description
      "Describes an external interface exposed by this VNF enabling
       connection with a Virual Link";
    leaf id {
      type string;
    }
    leaf virtual-link {
      tailf:info "Optional connection to an internal VLD";
      type leafref {
        path "../../virtual-link/id";
      }
    }
    leaf management {
      tailf:info "Can be set to indicate the management connection point for this VNF";
      type empty;
    }
  }
  list deployment-flavor {
    must "default-instantiation-level or count(instantiation-level) = 1";
    key "id";
    min-elements 1;
    description
      "Describes a specific deployment version of a VNF with specific
       requirements for capacity and performance.";
    reference "GS NFV IFA011: Information elements related to the DeploymentFlavour";
    leaf id {
      type string;
    }
    leaf description {
      tailf:info "Human readable description of the deployment flavor";
      type string;
```

```
}
list vdu-profile {
  must "min-number-of-instances<=max-number-of-instances";
  key "vdu";
  min-elements 1;
  description
    "The Vduprofile describes additional instantiation data for
     a given VDU used in a deployment flavour.";
  reference "GS NFV IFA011: VduProfile information element";
  leaf vdu {
    type leafref {
      path "../../../vdu/id";
    }
  }
  leaf min-number-of-instances {
    type uint16;
    mandatory true;
  }
  leaf max-number-of-instances {
    type uint16;
    mandatory true;
  }
}
list instantiation-level {
  key "id";
  min-elements 1;
  description
    "The InstantiationLevel information element describes a
     given level of resources to be instantiated within a
     deployment flavour in term of the number of VNFC instances
     to be created from each VDU.
     All the VDUs referenced in the level shall be part of the
     corresponding deployment flavour and their number shall
     be within the range (min/max) for this deployment flavour.";
  reference "GS NFV IFA011: InstantiationLevel information element";
  leaf id {
    type string;
  }
  leaf description {
    tailf:info "Human readable description of the instantiation level";
    type string;
  }
  list vdu-level {
    key "vdu";
    min-elements 1;
    description
      "Sets the number of instances for the VDU in this
       instantiation level.";
    leaf vdu {
      type leafref {
        path "../../../../vdu/id";
      }
    }
    leaf number-of-instances {
      type uint16;
      must ". <= ../../../vdu-profile[vdu=current()/../vdu]/max-number-of-instances";
      must ". >= ../../../vdu-profile[vdu=current()/../vdu]/min-number-of-instances";
      mandatory true;
    }
  }
}
leaf default-instantiation-level {
```

```
          type leafref {
            path "../instantiation-level/id";
          }
        }
      }
    }
  }
}
```

Example 16. NS YANG Models

```
submodule tailf-nfvo-ns {
  belongs-to tailf-nfvo {
    prefix nfvo;
  }

  include tailf-nfvo-common {
    revision-date 2016-03-11;
  }

  description
    "Models for NS according to ETSI GS IFA 014.";

  revision 2016-03-11 {
    description
      "Initial revision.";
    reference "ETSI GS IFA 014";
  }

  grouping nsd {
    list nsd {
      key "id";
      leaf id {
        type string;
        description
          "Uniquely identifies an Network Service Template";
      }
      leaf designer {
        type string;
        description
          "Identifies the designer of the network service";
      }
      leaf version {
        type string;
        description
          "Identifies the version of the network service";
      }
      list vnfd {
        key "vnfd";
        min-elements 1;
        description
          "This lists the VNFDs this network service consists of.";
        leaf vnfd {
          type leafref {
            path "/nfvo/vnfd/id";
          }
        }
        list connection-point {
          key "id";
          leaf id {
            type leafref {
              path "deref(../../vnfd)/../nfvo:external-connection-point/nfvo:id";
            }
```

```
        }
        choice ns-cp {
          mandatory true;
          description
            "A VNFDs connection point must either connect to an virtual
             link or a service access point.";
          leaf virtual-link {
            type leafref {
              path "../../../virtual-link/id";
            }
          }
          leaf service-access-point {
            type leafref {
              path "../../../service-access-point/id";
            }
          }
        }
      }
      list service-access-point {
        key "id";
        min-elements 1;
        description
          "A service access point serves as the network service's external
           connection point";
        leaf id {
          type string;
        }
        leaf description {
          type string;
        }
        leaf layer-protocol {
          type layer-protocol;
        }
        leaf virtual-link {
          type leafref {
            path "../../virtual-link/id";
          }
          description
            "Can optionally connect this service-access-point to an virtual-link. If not set,
             it's assumed a VNFD's connection-points references this
             service-access-point.";
        }
      }
      list virtual-link {
        key "id";
        description
          "The virtual links describes the networks service's internal networks";
        leaf id {
          type string;
        }
        leaf description {
          type string;
        }
        leaf layer-protocol {
          type layer-protocol;
          mandatory true;
        }
      }
    }
  }
}
```

Example 17. NFVO ESC YANG Models

```
module tailf-nfvo-esc {
  namespace "http://tail-f.com/pkg/tailf-nfvo-esc";
  prefix nfvo-esc;

  import ietf-inet-types { prefix inet; }
  import ietf-yang-types { prefix yang; }
  import tailf-common { prefix tailf; }
  import tailf-ncs { prefix ncs; }
  import tailf-nfvo { prefix nfvo; }
  import esc { prefix esc; }

  organization "Cisco Systems";

  contact "<support@tail-f.com>";

  description
    "The models for the ESC implementation of the VNF record types.";

  revision 2016-01-01 {
    description
      "Initial revision.";
  }

  identity vdu {
    description
      "Plan component used by the VNFR service";
    base ncs:plan-component-type;
  }

  identity deployed {
    description
      "Plan state used by the VNFR service";
    base ncs:plan-state;
  }

  grouping network-parameters {
    description
      "Used to specify networking parameters for virtual links";
    leaf dhcp {
      type empty;
    }

    container subnet {
      leaf network {
        mandatory true;
        type inet:ip-prefix;
      }

      choice gateway-choice {
        description
          "When creating a network, the gateway setting has three options:
           1. The VIM assigns gateway, in this case don't set this choice
           2. The VIM uses an existing gateway, set gateway
           3. No gateway should be used, set no-gateway";
        leaf gateway {
          type inet:ip-address;
        }
        leaf no-gateway {
          type empty;
        }
```

```
    }
  }
}

grouping vnfr-list {
  list vnfr {
    key id;

    leaf id {
      type leafref {
        path "/nfvo:nfvo/nfvo:vnfd/nfvo:id";
      }
    }

    leaf vnfd-flavor {
      mandatory true;
      type leafref {
        path "deref(../id)/../nfvo:deployment-flavor/nfvo:id";
      }
    }

    leaf instantiation-level {
      mandatory true;
      type leafref {
        path "deref(../vnfd-flavor)/../nfvo:instantiation-level/nfvo:id";
      }
    }

    leaf managed {
      description "If set, NSO will try to add all VDU instances to the device tree";
      type empty;
    }

    list vdu {
      key id;
      leaf id {
        type leafref {
          path "deref(../../id)/../nfvo:vdu/nfvo:id";
        }
      }

      leaf image-name {
        description
          "An image available in the VIM";
        type string;
        mandatory true;
      }

      leaf flavor-name {
        description
          "A flavor available in the VIM";
        type string;
        mandatory true;
      }

      must "count(/nfvo:nfvo/nfvo:vnfd[nfvo:id=current()/../id]/nfvo:vdu[nfvo:id=current()/
        error-message "All mandatory day0 destinations have to be set";
      }
      list day0 {
        key destination;
        description
          "Data for the VNF's day0.";
```

```
  leaf destination {
    description
      "Destination filename for this day0 file.";
    type string;
  }

  choice source {
    description
      "Specifies where ESC can get the day0 content";
    leaf url {
      type inet:uri;
    }

    leaf data {
      type string;
    }
  }

  list variable {
    description
      "Key/value store for the day0 data. ESC will substitute
       <$name> with <value> in the day0 file. ";
    key name;

    leaf name {
      type string;
    }

    leaf value {
      type string;
    }
  }
}

leaf authgroup {
  description
    "Points out the authentication group used if adding this VNF to the device
     tree";
  type leafref {
    path '/ncs:devices/ncs:authgroups/ncs:group/ncs:name';
  }
}

list connection-point-address {
  description
    "Possible to specify addresses for this connection point at
     instantiation. These addresses will be used at scaling.";
  key id;

  leaf id {
    type leafref {
      path "/nfvo:nfvo/nfvo:vnfd[nfvo:id=current()/../../../id]/nfvo:vdu[nfvo:id=current
    }
  }

  leaf address {
    description "If this address is set, it will be assigned to the interface,
                 otherwise the first address in the range will be selected.";
    type inet:ip-address;
  }
```

```
    leaf start {
      description "Range start";
      mandatory true;
      type inet:ip-address;
    }

    leaf end {
      description "Range end";
      mandatory true;
      type inet:ip-address;
    }
  }

  list device-template {
    ordered-by user;
    key name;
    description
      "After all settings has been made to ESC, but before commit, these
      templates are applied to the ESC devices. They are applied in the
      user specified order.

      This allows the user to make any change to the config written to the
      ESC.";

    leaf name {
      type leafref {
        path "/ncs:devices/ncs:template/ncs:name";
      }
    }
    list variable {
      description
        "Key/value store for the device template. The service will substitute
        <$name> with <value> in the device template.";
      key name;

      leaf name {
        type string;
      }

      leaf value {
        type string;
      }
    }
  }
}

list virtual-link {
  key id;
  description
    "Possible to set the network parameters for the VNFD's virtual links";

  leaf id {
    type leafref {
      path "deref(../../id)/../nfvo:virtual-link/nfvo:id";
    }
  }

  uses network-parameters;
}

list vnfd-connection-point {
  key id;
```

```
      description "Connects the VNFD's connection points to VLRs";

      leaf id {
        type leafref {
          path "deref(../../id)/../nfvo:external-connection-point/nfvo:id";
        }
      }

      leaf vlr {
        mandatory true;
        type leafref {
          tailf:no-leafref-check;
          path "/nfvo:nfvo/nfvo:vlr/nfvo-esc:esc/nfvo-esc:vlr/nfvo-esc:id";
        }
      }
    }
  }
}

grouping vnf-deployment {
  leaf tenant {
    type leafref {
      path "/ncs:devices/ncs:device/ncs:config/esc:esc_datamodel/esc:tenants/esc:tenant/esc:na
    }
  }

  leaf deployment-name {
    description "A per tenant unique deployment name";
    type string;
  }

  leaf username {
    description
      "Authenticated user for invoking the service.
       The system will use this user in all interactions with CDB
       to honor AAA rules";
    tailf:info "Authenticated user for invoking the service.";
    type string;
    mandatory true;
  }

  list esc-device {
    key name;

    leaf name {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }
  }

  leaf staged-delete {
    description
      "If set; staged delete will be used. Please note that this
       requires a staged delete package to manage this, see documentation
       for details";
    tailf:info "Enable staged delete for this deployment";
    type empty;
  }

  uses vnfr-list;
  list vlr {
```

```
    description
      "Lists the VLRs the VNFR should create as part of the deployment.";
    key id;
    leaf id {
      mandatory true;
      type leafref {
        tailf:no-leafref-check;
        path "/nfvo:nfvo/nfvo:vlr/nfvo-esc:esc/nfvo-esc:vlr/nfvo-esc:id";
      }
    }
  }
}

augment /nfvo:nfvo/nfvo:vnfr {
  container esc {
    list vnf-deployment {
      uses ncs:plan-data;
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-esc-vnfr;

      key "tenant deployment-name";
      uses vnf-deployment;
      description
        "This is the interface where NB services writes VNFR. This in it's
         turn invokes the real VNFR service through a subscriber.

         This allows for staged delete.";

      must "count(/nfvo:nfvo/nfvo-esc:settings-esc/netconf-subscription/esc-device[name=cur
        error-message "NETCONF subcriptions has to be setup in /nfvo/settings-esc/netconf-s
      }
    }

    list vnf-deployment-result {
      description "we'll store the lookup from esc vmid to a
        index here. this will enable us to generate
        device names when scaling in/out";
      config false;
      tailf:cdb-oper {
        tailf:persistent true;
      }
      key "tenant deployment-name";
      leaf tenant {
        type string;
      }

      leaf deployment-name {
        type string;
      }

      container status {
        choice cstatus {
          leaf deployed {
            type empty;
          }
          leaf ready {
            type yang:date-and-time;
          }
          leaf error {
            type string;
          }
        }
```

```
      }

      list vdu {
        key "vnfr vdu";
        leaf vnfr {
          type string;
        }

        leaf vdu {
          type string;
        }
        list vm-device {
          key "esc-device vmid";
          leaf esc-device {
            type string;
          }

          leaf vmid {
            type string;
          }

          leaf index {
            type uint16;
          }

          leaf device-name {
            type string;
          }

          container status {
            choice cstatus {
              leaf deployed {
                type empty;
              }
              leaf ready {
                type yang:date-and-time;
              }
              leaf error {
                type string;
              }
              leaf recovering {
                type yang:date-and-time;
              }
            }
          }
        }
      }
    }
  }
}

augment /nfvo:nfvo/nfvo:vlr {
  container esc {
    list vlr {
      key id;

      leaf id {
        type string;
      }

      choice source {
        leaf vld {
```

```
            type leafref {
              path "/nfvo:nfvo/nfvo:vld/nfvo:id";
            }
          }
          container nsr {
            leaf tenant {
              type leafref {
                path "/nfvo:nfvo/nfvo:nsr/nfvo-esc:esc/nfvo-esc:nsr/nfvo-esc:tenant";
              }
            }

            leaf deployment-name {
              description "A per tenant unique deployment name";
              type leafref {
                path "deref(../tenant)/../nfvo-esc:deployment-name";
              }
            }

            leaf virtual-link {
              type leafref {
                path "deref(../deployment-name)/../nfvo-esc:virtual-link/nfvo-esc:id";
              }
            }
          }
        }

        container network-parameters {
          presence true;
          uses network-parameters;
        }
      }
    }
  }

  augment /nfvo:nfvo/nfvo:nsr {
    container esc {
      list nsr {
        uses ncs:plan-data;
        uses ncs:service-data;
        ncs:servicepoint tailf-nfvo-esc-nsr;
        key "tenant deployment-name";

        description
          "The NSR service";

        leaf tenant {
          type leafref {
            path "/ncs:devices/ncs:device/ncs:config/esc:esc_datamodel/esc:tenants/esc:tenant
          }
        }

        leaf deployment-name {
          description "A per tenant unique deployment name";
          type string;
        }

        leaf username {
          description
            "Authenticated user for invoking the service";
          type string;
          mandatory true;
        }
```

```
leaf nsd {
  mandatory true;
  type leafref {
    path "/nfvo:nfvo/nfvo:nsd/nfvo:id";
  }
}

list esc-device {
  key name;

  leaf name {
    type leafref {
      path "/ncs:devices/ncs:device/ncs:name";
    }
  }
}

leaf staged-delete {
  description
    "If set; staged delete will be used. Please note that this
    requires a staged delete package to manage this, see documentation
    for details";
  tailf:info "Enable staged delete for this deployment";
  type empty;
}

uses vnfr-list;

list virtual-link {
  description
    "Networking parameters for the NSD virtual links";
  key id;

  leaf id {
    type leafref {
      path "deref(../../nsd)/../nfvo:virtual-link/nfvo:id";
    }
  }

  uses network-parameters;
}

list service-access-point {
  key id;

  description
    "Maps the NSD's SAPs to VLRs";

  leaf id {
    type leafref {
      path "deref(../../nsd)/../nfvo:service-access-point/nfvo:id";
    }
  }

  leaf vlr {
    mandatory true;
    type leafref {
      path "/nfvo:nfvo/nfvo:vlr/nfvo-esc:esc/nfvo-esc:vlr/nfvo-esc:id";
    }
  }
}
```

```
    }
  }
}

augment /nfvo:nfvo/nfvo:onboarding {
  container esc {
    list flavor {
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-esc-onboard-flavor;

      description
        "Service to create flavor from VDUs in ESC";

      key "name";
      leaf name {
        type string;
      }

      leaf vnfd {
        mandatory true;
        type leafref {
          path "/nfvo:nfvo/nfvo:vnfd/nfvo:id";
        }
      }

      leaf vdu {
        mandatory true;
        type leafref {
          path "deref(../vnfd)/../nfvo:vdu/nfvo:id";
        }
      }

      list esc-device {
        key name;

        leaf name {
          type leafref {
            path "/ncs:devices/ncs:device/ncs:name";
          }
        }
      }
    }

    list image {
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-esc-onboard-image;

      description
        "Service to create flavor from VDUs in ESC";

      key "name";
      leaf name {
        type string;
      }

      leaf vnfd {
        mandatory true;
        type leafref {
          path "/nfvo:nfvo/nfvo:vnfd/nfvo:id";
        }
      }
```

```
        leaf vdu {
          mandatory true;
          type leafref {
            path "deref(../vnfd)/../nfvo:vdu/nfvo:id";
          }
        }

        must "/nfvo:nfvo/nfvo:vnfd[nfvo:id=current()/vnfd]/nfvo:vdu[nfvo:id=current()/vdu]/nfvo:
          error-message "Image has to be specified";
        }

        list esc-device {
          key name;

          leaf name {
            type leafref {
              path "/ncs:devices/ncs:device/ncs:name";
            }
          }
        }
      }
    }
  }

  augment /nfvo:nfvo {
    container tailf-esc-internal {
      list vnf-deployment {
        uses ncs:service-data;
        ncs:servicepoint tailf-nfvo-esc-vnfr-backend;
        key "tenant deployment-name";
        uses vnf-deployment;
        description "The actual VNFR service instance.";
      }
    }

    container settings-esc {
      presence true;
      uses ncs:service-data;
      ncs:servicepoint tailf-nfvo-esc-settings;

      container netconf-subscription {
        leaf username {
          tailf:info "Username used to setup NETCONF subscriptions to the ESC devices";
          type string;
          mandatory true;
        }

        list esc-device {
          key name;

          leaf name {
            type leafref {
              path "/ncs:devices/ncs:device/ncs:name";
            }
          }
        }
      }
    }
  }
}
```

Example 18. Staged Delete Model

```
module tailf-nfvo-esc-staged-delete {
  namespace "http://tail-f.com/pkg/tailf-nfvo-esc-staged-delete";
  prefix nfvo-esc-sd;

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

  organization "Cisco Systems";

  contact "<support@tail-f.com>";

  description
    "This model is used by the NFVE ESC to call out
    to a custom handler for e.g. revoking a license of a device
    that the VNFR want to remove.

    To initiate the delete of a license, the VNFR add
    the device name to the 'delete' container.

    The custom handler must subscribe to changes in the 'delete'
    container and act accordingly when new devices are added.
    When the license has been deleted, the 'device' is removed
    from the 'delete' container and inserted in the 'deleted' container.

    The VNFR subscribes to changes in the 'deleted'
    container and will, when triggered, pick up the device in order
    to remove it; and will delete the device from the 'deleted' container.";

  revision 2016-04-08 {
    description "License handling";
  }

  grouping sd-grp {
    leaf tenant {
      description "Name of the tenant to be removed";
      type string;
    }
    leaf deployment-name {
      description "Name of the deployment name to be removed";
      type string;
    }
    leaf username {
      description "Username of the ESC service";
      mandatory true;
      type string;
    }
  }

  augment /nfvo:nfvo {
    container staged-delete-esc {
      list delete {
        key "tenant deployment-name";
        uses sd-grp;
      }

      list deleted {
        key "tenant deployment-name";
        uses sd-grp;
      }
    }
```

```
  }
}
```

Example 19. NFVO

```
module: tailf-nfvo
   +--rw nfvo
      +--rw vld* [id]
      |  +--rw id             string
      |  +--rw vendor?        string
      |  +--rw description?   string
      |  +--rw version?       string
      +--rw vnfd* [id]
      |  +--rw id                           string
      |  +--rw name?                        string
      |  +--rw provider?                    string
      |  +--rw version?                     string
      |  +--rw description?                 string
      |  +--rw vdu* [id]
      |  |  +--rw id                           string
      |  |  +--rw description?                 string
      |  |  +--rw internal-connection-point* [id]
      |  |  |  +--rw id                           string
      |  |  |  +--rw description?                 string
      |  |  |  +--rw (cp-connection)
      |  |  |  |  +--:(virtual-link)
      |  |  |  |  |  +--rw virtual-link?               -> ../../../virtual-link/id
      |  |  |  |  +--:(external-connection-point)
      |  |  |  |     +--rw external-connection-point?   -> ../../../external-connection-point/id
      |  |  |  +--rw interface-id                 uint8
      |  |  +--rw virtual-compute
      |  |  |  +--rw virtual-memory
      |  |  |  |  +--rw virtual-memory-size    decimal64
      |  |  |  +--rw virtual-cpu
      |  |  |     +--rw number-of-virtual-cpus    uint16
      |  |  +--rw virtual-storage* [id]
      |  |  |  +--rw id                 string
      |  |  |  +--rw type-of-storage    enumeration
      |  |  |  +--rw size-of-storage    uint64
      |  |  +--rw software-image!
      |  |  |  +--rw name?                string
      |  |  |  +--rw container-format     enumeration
      |  |  |  +--rw disk-format          enumeration
      |  |  |  +--rw image                inet:uri
      |  |  |  +--rw additional-setting* [id]
      |  |  |     +--rw id       string
      |  |  |     +--rw value?   string
      |  |  +--rw device-type
      |  |  |  +--rw (ne-type)?
      |  |  |     +--:(netconf)
      |  |  |     |  +--rw netconf?   empty
      |  |  |     +--:(generic)
      |  |  |     |  +--rw generic!
      |  |  |     |     +--rw ned-id    string
      |  |  |     +--:(cli)
      |  |  |        +--rw cli!
      |  |  |           +--rw ned-id      string
      |  |  |           +--rw protocol?   enumeration
      |  |  |           +--rw port?       inet:port-number
      |  |  +--rw depends-on* [vdu]
      |  |  |  +--rw vdu    -> ../../../vdu/id
      |  |  +--rw bootup-time                  uint64
      |  |  +--rw recovery-wait-time           uint64
```

```
|  |  +--rw day0* [destination]
|  |     +--rw destination    string
|  |     +--rw mandatory?     empty
|  +--rw virtual-link* [id]
|  |  +--rw id                string
|  |  +--rw description?      string
|  |  +--rw layer-protocol    layer-protocol
|  +--rw external-connection-point* [id]
|  |  +--rw id              string
|  |  +--rw virtual-link?   -> ../../virtual-link/id
|  |  +--rw management?     empty
|  +--rw deployment-flavor* [id]
|     +--rw id                              string
|     +--rw description?                    string
|     +--rw vdu-profile* [vdu]
|     |  +--rw vdu                          -> ../../../vdu/id
|     |  +--rw min-number-of-instances      uint16
|     |  +--rw max-number-of-instances      uint16
|     +--rw instantiation-level* [id]
|     |  +--rw id             string
|     |  +--rw description?   string
|     |  +--rw vdu-level* [vdu]
|     |     +--rw vdu                     -> ../../../../vdu/id
|     |     +--rw number-of-instances     uint16
|     +--rw default-instantiation-level?   -> ../instantiation-level/id
+--rw nsd* [id]
|  +--rw id                      string
|  +--rw designer?               string
|  +--rw version?                string
|  +--rw vnfd* [vnfd]
|  |  +--rw vnfd                -> /nfvo/vnfd/id
|  |  +--rw connection-point* [id]
|  |     +--rw id                        -> deref(../../vnfd)/../external-connection-poin
|  |     +--rw (ns-cp)
|  |        +--:(virtual-link)
|  |        |  +--rw virtual-link?           -> ../../../virtual-link/id
|  |        +--:(service-access-point)
|  |           +--rw service-access-point?   -> ../../../service-access-point/id
|  +--rw service-access-point* [id]
|  |  +--rw id                string
|  |  +--rw description?      string
|  |  +--rw layer-protocol?   layer-protocol
|  |  +--rw virtual-link?     -> ../../virtual-link/id
|  +--rw virtual-link* [id]
|     +--rw id                string
|     +--rw description?      string
|     +--rw layer-protocol    layer-protocol
+--rw vnfr
+--rw vlr
+--rw nsr
+--rw onboarding
```

Example 20. NFVO ESC

```
module: tailf-nfvo-esc
augment /nfvo:nfvo/nfvo:vnfr:
   +--rw esc
      +--rw vnf-deployment* [tenant deployment-name]
      |  +--ro plan
      |  |  +--ro component* [name]
      |  |  |  +--ro name     string
      |  |  |  +--ro type     plan-component-type-t
      |  |  |  +--ro state* [name]
```

```
|  |  |     +--ro name      plan-state-name-t
|  |  |     +--ro status    plan-state-status-t
|  |  |     +--ro when?     yang:date-and-time
|  |  +--ro failed?      empty
|  +--ro plan-history
|  |  +--ro plan* [time]
|  |     +--ro time         yang:date-and-time
|  |     +--ro component* [name]
|  |        +--ro name     string
|  |        +--ro type     plan-component-type-t
|  |        +--ro state* [name]
|  |           +--ro name      plan-state-name-t
|  |           +--ro status    plan-state-status-t
|  |           +--ro when?     yang:date-and-time
|  +---x check-sync
|  |  +---w input
|  |  |  +---w outformat?                 outformat4
|  |  |  +---w (depth)?
|  |  |  |  +--:(deep)
|  |  |  |  |  +---w deep?                      empty
|  |  |  |  +--:(shallow)
|  |  |  |     +---w shallow?                   empty
|  |  |  +---w suppress-positive-result?   empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |        +--:(in-sync)
|  |        |  +--ro in-sync?      boolean
|  |        +--:(result-xml)
|  |        |  +--ro result-xml
|  |        +--:(cli)
|  |        |  +--ro cli?          string
|  |        +--:(native)
|  |           +--ro native
|  |              +--ro device*
|  |                 +--ro name?   string
|  |                 +--ro data?   string
|  +---x deep-check-sync
|  |  +---w input
|  |  |  +---w outformat?                 outformat-deep-check-sync
|  |  |  +---w suppress-positive-result?   empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |        +--:(in-sync)
|  |        |  +--ro in-sync?      boolean
|  |        +--:(result-xml)
|  |        |  +--ro result-xml
|  |        +--:(cli)
|  |           +--ro cli?          string
|  +---x re-deploy
|  |  +---w input
|  |  |  +---w dry-run!
|  |  |  |  +---w outformat?   outformat3
|  |  |  +---w no-revision-drop?       empty
|  |  |  +---w no-overwrite?           empty
|  |  |  +---w no-networking?          empty
|  |  |  +---w no-out-of-sync-check?   empty
|  |  |  +---w commit-queue!
|  |  |  |  +---w tag?            string
|  |  |  |  +---w (operation-mode)?
|  |  |  |  |  +--:(async)
|  |  |  |  |  |  +---w async?          empty
|  |  |  |  |  +--:(sync)
```

```
|  |  |  |  |  |  +---w sync!
|  |  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |  |        +--:(timeout)
|  |  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |  |        +--:(infinity)
|  |  |  |  |  |           +---w infinity?   empty
|  |  |  |  |  +--:(bypass)
|  |  |  |  |     +---w bypass?         empty
|  |  |  |  +---w block-others?   empty
|  |  |  |  +---w lock?           empty
|  |  |  +---w (depth)?
|  |  |  |  +--:(deep)
|  |  |  |  |  +---w deep?                   empty
|  |  |  |  +--:(shallow)
|  |  |  |     +---w shallow?                empty
|  |  |  +---w reconcile!
|  |  |     +---w (c-non-service-config)?
|  |  |        +--:(keep-non-service-config)
|  |  |        |  +---w keep-non-service-config?      empty
|  |  |        +--:(discard-non-service-config)
|  |  |           +---w discard-non-service-config?   empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |     |  +--:(result-xml)
|  |     |  |  +--ro result-xml
|  |     |  +--:(cli)
|  |     |  |  +--ro cli?            string
|  |     |  +--:(native)
|  |     |     +--ro native
|  |     |        +--ro device*
|  |     |           +--ro name?   string
|  |     |           +--ro data?   string
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
|  |        +--ro status?   enumeration
|  +---x reactive-re-deploy
|  |  +--ro output
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
|  |        +--ro status?   enumeration
|  +---x touch
|  x--ro device-modifications?       string
|  +--ro modified
|  |  +--ro devices*    -> /ncs:devices/device/name
|  |  +--ro services*   instance-identifier
|  +--ro directly-modified
|  |  +--ro devices*    -> /ncs:devices/device/name
|  |  +--ro services*   instance-identifier
|  +---x get-modifications
|  |  +---w input
|  |  |  +---w outformat?   outformat2
|  |  |  +---w reverse?     empty
|  |  |  +---w (depth)?
|  |  |     +--:(deep)
|  |  |     |  +---w deep?        empty
|  |  |     +--:(shallow)
|  |  |        +---w shallow?     empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |        +--:(cli)
```

```
|  |        |  +--ro cli?          string
|  |        +--:(result-xml)
|  |           +--ro result-xml
|  x--ro device-list*                string
|  +---x un-deploy
|  |  +---w input
|  |  |  +---w no-revision-drop?       empty
|  |  |  +---w no-overwrite?           empty
|  |  |  +---w no-networking?          empty
|  |  |  +---w no-out-of-sync-check?   empty
|  |  |  +---w commit-queue!
|  |  |  |  +---w tag?            string
|  |  |  |  +---w (operation-mode)?
|  |  |  |  |  +--:(async)
|  |  |  |  |  |  +---w async?          empty
|  |  |  |  |  +--:(sync)
|  |  |  |  |  |  +---w sync!
|  |  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |  |        +--:(timeout)
|  |  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |  |        +--:(infinity)
|  |  |  |  |  |           +---w infinity?   empty
|  |  |  |  |  +--:(bypass)
|  |  |  |  |     +---w bypass?         empty
|  |  |  |  +---w block-others?   empty
|  |  |  |  +---w lock?           empty
|  |  |  +---w ignore-refcount?        empty
|  |  +--ro output
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
|  |        +--ro status?   enumeration
|  +--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
|  +--ro commit-queue
|  |  +--ro status?                    enumeration
|  |  +--ro cleared-by-admin?          empty
|  |  +--ro no-of-impacted-services?   uint32
|  |  +--ro failed-device* [name]
|  |  |  +--ro name           -> /ncs:devices/device/name
|  |  |  +--ro time?          yang:date-and-time
|  |  |  +--ro config-data?   string
|  |  |  +--ro error?         string
|  |  +--ro queue-item?               uint64
|  |  +---x clear
|  |  +---x purge
|  +--rw private
|  |  +--rw diff-set?              binary
|  |  +--rw forward-diff-set?      binary
|  |  +--rw device-list*           string
|  |  +--rw service-list*          instance-identifier
|  |  +--rw property-list
|  |  |  +--rw property* [name]
|  |  |     +--rw name     string
|  |  |     +--rw value?   string
|  |  +--rw re-deploy-counter?     int32
|  |  +--rw latest-commit-params?  binary
|  |  +--rw latest-u-info?         binary
|  +--rw tenant                     -> /ncs:devices/device/config/esc:esc_datamodel/tenan
|  +--rw deployment-name            string
|  +--rw username                   string
|  +--rw esc-device* [name]
|  |  +--rw name    -> /ncs:devices/device/name
```

```
|  +--rw staged-delete?             empty
|  +--rw vnfr* [id]
|  |  +--rw id                      -> /nfvo:nfvo/vnfd/id
|  |  +--rw vnfd-flavor             -> deref(../id)/../nfvo:deployment-flavor/id
|  |  +--rw instantiation-level     -> deref(../vnfd-flavor)/../nfvo:instantiation-leve
|  |  +--rw managed?                empty
|  |  +--rw vdu* [id]
|  |  |  +--rw id                         -> deref(../../id)/../nfvo:vdu/id
|  |  |  +--rw image-name                 string
|  |  |  +--rw flavor-name                string
|  |  |  +--rw day0* [destination]
|  |  |  |  +--rw destination    string
|  |  |  |  +--rw (source)?
|  |  |  |  |  +--:(url)
|  |  |  |  |  |  +--rw url?           inet:uri
|  |  |  |  |  +--:(data)
|  |  |  |  |     +--rw data?          string
|  |  |  |  +--rw variable* [name]
|  |  |  |     +--rw name     string
|  |  |  |     +--rw value?   string
|  |  |  +--rw authgroup?                 -> /ncs:devices/authgroups/group/name
|  |  |  +--rw connection-point-address* [id]
|  |  |  |  +--rw id         -> /nfvo:nfvo/vnfd[nfvo:id=current()/../../../id]/vdu[nfvo
|  |  |  |  +--rw address?   inet:ip-address
|  |  |  |  +--rw start      inet:ip-address
|  |  |  |  +--rw end        inet:ip-address
|  |  |  +--rw device-template* [name]
|  |  |     +--rw name        -> /ncs:devices/template/name
|  |  |     +--rw variable* [name]
|  |  |        +--rw name     string
|  |  |        +--rw value?   string
|  |  +--rw virtual-link* [id]
|  |  |  +--rw id        -> deref(../../id)/../nfvo:virtual-link/id
|  |  |  +--rw dhcp?     empty
|  |  |  +--rw subnet
|  |  |     +--rw network       inet:ip-prefix
|  |  |     +--rw (gateway-choice)?
|  |  |        +--:(gateway)
|  |  |        |  +--rw gateway?      inet:ip-address
|  |  |        +--:(no-gateway)
|  |  |           +--rw no-gateway?   empty
|  |  +--rw vnfd-connection-point* [id]
|  |     +--rw id     -> deref(../../id)/../nfvo:external-connection-point/id
|  |     +--rw vlr    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
|  +--rw vlr* [id]
|     +--rw id    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
+--ro vnf-deployment-result* [tenant deployment-name]
   +--ro tenant             string
   +--ro deployment-name    string
   +--ro status
   |  +--ro (cstatus)?
   |     +--:(deployed)
   |     |  +--ro deployed?   empty
   |     +--:(ready)
   |     |  +--ro ready?      yang:date-and-time
   |     +--:(error)
   |        +--ro error?      string
   +--ro vdu* [vnfr vdu]
      +--ro vnfr         string
      +--ro vdu          string
      +--ro vm-device* [esc-device vmid]
         +--ro esc-device     string
```

```
               +--ro vmid           string
               +--ro index?         uint16
               +--ro device-name?   string
               +--ro status
                  +--ro (cstatus)?
                     +--:(deployed)
                     |  +--ro deployed?     empty
                     +--:(ready)
                     |  +--ro ready?        yang:date-and-time
                     +--:(error)
                     |  +--ro error?        string
                     +--:(recovering)
                        +--ro recovering?   yang:date-and-time
augment /nfvo:nfvo/nfvo:vlr:
   +--rw esc
      +--rw vlr* [id]
         +--rw id                    string
         +--rw (source)?
         |  +--:(vld)
         |  |  +--rw vld?                 -> /nfvo:nfvo/vld/id
         |  +--:(nsr)
         |     +--rw nsr
         |        +--rw tenant?            -> /nfvo:nfvo/nsr/nfvo-esc:esc/nsr/tenant
         |        +--rw deployment-name?   -> deref(../tenant)/../deployment-name
         |        +--rw virtual-link?      -> deref(../deployment-name)/../virtual-link/id
         +--rw network-parameters!
            +--rw dhcp?     empty
            +--rw subnet
               +--rw network       inet:ip-prefix
               +--rw (gateway-choice)?
                  +--:(gateway)
                  |  +--rw gateway?      inet:ip-address
                  +--:(no-gateway)
                     +--rw no-gateway?   empty
augment /nfvo:nfvo/nfvo:nsr:
   +--rw esc
      +--rw nsr* [tenant deployment-name]
         +--ro plan
         |  +--ro component* [name]
         |  |  +--ro name     string
         |  |  +--ro type     plan-component-type-t
         |  |  +--ro state* [name]
         |  |     +--ro name      plan-state-name-t
         |  |     +--ro status    plan-state-status-t
         |  |     +--ro when?     yang:date-and-time
         |  +--ro failed?      empty
         +--ro plan-history
         |  +--ro plan* [time]
         |     +--ro time         yang:date-and-time
         |     +--ro component* [name]
         |        +--ro name     string
         |        +--ro type     plan-component-type-t
         |        +--ro state* [name]
         |           +--ro name      plan-state-name-t
         |           +--ro status    plan-state-status-t
         |           +--ro when?     yang:date-and-time
         +---x check-sync
         |  +---w input
         |  |  +---w outformat?                 outformat4
         |  |  +---w (depth)?
         |  |  |  +--:(deep)
         |  |  |  |  +---w deep?                      empty
```

```
|  |  |  +--:(shallow)
|  |  |     +---w shallow?                    empty
|  |  +---w suppress-positive-result?   empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(in-sync)
|        |  +--ro in-sync?      boolean
|        +--:(result-xml)
|        |  +--ro result-xml
|        +--:(cli)
|        |  +--ro cli?          string
|        +--:(native)
|           +--ro native
|              +--ro device*
|                 +--ro name?   string
|                 +--ro data?   string
+---x deep-check-sync
|  +---w input
|  |  +---w outformat?                  outformat-deep-check-sync
|  |  +---w suppress-positive-result?   empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(in-sync)
|        |  +--ro in-sync?      boolean
|        +--:(result-xml)
|        |  +--ro result-xml
|        +--:(cli)
|           +--ro cli?          string
+---x re-deploy
|  +---w input
|  |  +---w dry-run!
|  |  |  +---w outformat?   outformat3
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
|  |  +---w (depth)?
|  |  |  +--:(deep)
|  |  |  |  +---w deep?                   empty
|  |  |  +--:(shallow)
|  |  |     +---w shallow?                empty
|  |  +---w reconcile!
|  |     +---w (c-non-service-config)?
|  |        +--:(keep-non-service-config)
|  |        |  +---w keep-non-service-config?      empty
|  |        +--:(discard-non-service-config)
```

```
|  |          +---w discard-non-service-config?   empty
|  +--ro output
|     +--ro (outformat)?
|     |  +--:(result-xml)
|     |  |  +--ro result-xml
|     |  +--:(cli)
|     |  |  +--ro cli?            string
|     |  +--:(native)
|     |     +--ro native
|     |        +--ro device*
|     |           +--ro name?   string
|     |           +--ro data?   string
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x reactive-re-deploy
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x touch
x--ro device-modifications?       string
+--ro modified
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+--ro directly-modified
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+---x get-modifications
|  +---w input
|  |  +---w outformat?   outformat2
|  |  +---w reverse?     empty
|  |  +---w (depth)?
|  |     +--:(deep)
|  |     |  +---w deep?        empty
|  |     +--:(shallow)
|  |        +---w shallow?     empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(cli)
|        |  +--ro cli?          string
|        +--:(result-xml)
|           +--ro result-xml
x--ro device-list*                string
+---x un-deploy
|  +---w input
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
```

```
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
|  |  +---w ignore-refcount?        empty
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
+--ro commit-queue
|  +--ro status?                    enumeration
|  +--ro cleared-by-admin?          empty
|  +--ro no-of-impacted-services?   uint32
|  +--ro failed-device* [name]
|  |  +--ro name           -> /ncs:devices/device/name
|  |  +--ro time?          yang:date-and-time
|  |  +--ro config-data?   string
|  |  +--ro error?         string
|  +--ro queue-item?                uint64
|  +---x clear
|  +---x purge
+--rw private
|  +--rw diff-set?              binary
|  +--rw forward-diff-set?      binary
|  +--rw device-list*           string
|  +--rw service-list*          instance-identifier
|  +--rw property-list
|  |  +--rw property* [name]
|  |     +--rw name     string
|  |     +--rw value?   string
|  +--rw re-deploy-counter?     int32
|  +--rw latest-commit-params?   binary
|  +--rw latest-u-info?         binary
+--rw tenant                     -> /ncs:devices/device/config/esc:esc_datamodel/te
+--rw deployment-name            string
+--rw username                   string
+--rw nsd                        -> /nfvo:nfvo/nsd/id
+--rw esc-device* [name]
|  +--rw name    -> /ncs:devices/device/name
+--rw staged-delete?             empty
+--rw vnfr* [id]
|  +--rw id                      -> /nfvo:nfvo/vnfd/id
|  +--rw vnfd-flavor             -> deref(../id)/../nfvo:deployment-flavor/id
|  +--rw instantiation-level     -> deref(../vnfd-flavor)/../nfvo:instantiation-lev
|  +--rw managed?                empty
|  +--rw vdu* [id]
|  |  +--rw id                         -> deref(../../id)/../nfvo:vdu/id
|  |  +--rw image-name                 string
|  |  +--rw flavor-name                string
|  |  +--rw day0* [destination]
|  |  |  +--rw destination    string
|  |  |  +--rw (source)?
|  |  |  |  +--:(url)
|  |  |  |  |  +--rw url?          inet:uri
|  |  |  |  +--:(data)
|  |  |  |     +--rw data?         string
|  |  |  +--rw variable* [name]
|  |  |     +--rw name     string
```

```
|  |  |     +--rw value?   string
|  |  +--rw authgroup?                  -> /ncs:devices/authgroups/group/name
|  |  +--rw connection-point-address* [id]
|  |  |  +--rw id         -> /nfvo:nfvo/vnfd[nfvo:id=current()/../../../id]/vdu[nfvo:id
|  |  |  +--rw address?   inet:ip-address
|  |  |  +--rw start      inet:ip-address
|  |  |  +--rw end        inet:ip-address
|  |  +--rw device-template* [name]
|  |     +--rw name        -> /ncs:devices/template/name
|  |     +--rw variable* [name]
|  |        +--rw name     string
|  |        +--rw value?   string
|  +--rw virtual-link* [id]
|  |  +--rw id        -> deref(../../id)/../nfvo:virtual-link/id
|  |  +--rw dhcp?     empty
|  |  +--rw subnet
|  |     +--rw network       inet:ip-prefix
|  |     +--rw (gateway-choice)?
|  |        +--:(gateway)
|  |        |  +--rw gateway?      inet:ip-address
|  |        +--:(no-gateway)
|  |           +--rw no-gateway?   empty
|  +--rw vnfd-connection-point* [id]
|     +--rw id     -> deref(../../id)/../nfvo:external-connection-point/id
|     +--rw vlr    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
+--rw virtual-link* [id]
|  +--rw id        -> deref(../../nsd)/../nfvo:virtual-link/id
|  +--rw dhcp?     empty
|  +--rw subnet
|     +--rw network       inet:ip-prefix
|     +--rw (gateway-choice)?
|        +--:(gateway)
|        |  +--rw gateway?      inet:ip-address
|        +--:(no-gateway)
|           +--rw no-gateway?   empty
+--rw service-access-point* [id]
   +--rw id     -> deref(../../nsd)/../nfvo:service-access-point/id
   +--rw vlr    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
augment /nfvo:nfvo/nfvo:onboarding:
   +--rw esc
      +--rw flavor* [name]
      |  +---x check-sync
      |  |  +---w input
      |  |  |  +---w outformat?                 outformat4
      |  |  |  +---w (depth)?
      |  |  |  |  +--:(deep)
      |  |  |  |  |  +---w deep?                      empty
      |  |  |  |  +--:(shallow)
      |  |  |  |     +---w shallow?                   empty
      |  |  |  +---w suppress-positive-result?   empty
      |  |  +--ro output
      |  |     +--ro (outformat)?
      |  |        +--:(in-sync)
      |  |        |  +--ro in-sync?      boolean
      |  |        +--:(result-xml)
      |  |        |  +--ro result-xml
      |  |        +--:(cli)
      |  |        |  +--ro cli?          string
      |  |        +--:(native)
      |  |           +--ro native
      |  |              +--ro device*
      |  |                 +--ro name?   string
```

```
|  |                +--ro data?   string
|  +---x deep-check-sync
|  |  +---w input
|  |  |  +---w outformat?                  outformat-deep-check-sync
|  |  |  +---w suppress-positive-result?   empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |        +--:(in-sync)
|  |        |  +--ro in-sync?      boolean
|  |        +--:(result-xml)
|  |        |  +--ro result-xml
|  |        +--:(cli)
|  |           +--ro cli?          string
|  +---x re-deploy
|  |  +---w input
|  |  |  +---w dry-run!
|  |  |  |  +---w outformat?   outformat3
|  |  |  +---w no-revision-drop?       empty
|  |  |  +---w no-overwrite?           empty
|  |  |  +---w no-networking?          empty
|  |  |  +---w no-out-of-sync-check?   empty
|  |  |  +---w commit-queue!
|  |  |  |  +---w tag?            string
|  |  |  |  +---w (operation-mode)?
|  |  |  |  |  +--:(async)
|  |  |  |  |  |  +---w async?          empty
|  |  |  |  |  +--:(sync)
|  |  |  |  |  |  +---w sync!
|  |  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |  |        +--:(timeout)
|  |  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |  |        +--:(infinity)
|  |  |  |  |  |           +---w infinity?   empty
|  |  |  |  |  +--:(bypass)
|  |  |  |  |     +---w bypass?         empty
|  |  |  |  +---w block-others?   empty
|  |  |  |  +---w lock?           empty
|  |  |  +---w (depth)?
|  |  |  |  +--:(deep)
|  |  |  |  |  +---w deep?                  empty
|  |  |  |  +--:(shallow)
|  |  |  |     +---w shallow?               empty
|  |  |  +---w reconcile!
|  |  |     +---w (c-non-service-config)?
|  |  |        +--:(keep-non-service-config)
|  |  |        |  +---w keep-non-service-config?      empty
|  |  |        +--:(discard-non-service-config)
|  |  |           +---w discard-non-service-config?   empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |     |  +--:(result-xml)
|  |     |  |  +--ro result-xml
|  |     |  +--:(cli)
|  |     |  |  +--ro cli?            string
|  |     |  +--:(native)
|  |     |     +--ro native
|  |     |        +--ro device*
|  |     |           +--ro name?   string
|  |     |           +--ro data?   string
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
```

```
|  |       +--ro status?   enumeration
|  +---x reactive-re-deploy
|  |  +--ro output
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
|  |        +--ro status?   enumeration
|  +---x touch
|  x--ro device-modifications?       string
|  +--ro modified
|  |  +--ro devices*    -> /ncs:devices/device/name
|  |  +--ro services*   instance-identifier
|  +--ro directly-modified
|  |  +--ro devices*    -> /ncs:devices/device/name
|  |  +--ro services*   instance-identifier
|  +---x get-modifications
|  |  +---w input
|  |  |  +---w outformat?   outformat2
|  |  |  +---w reverse?     empty
|  |  |  +---w (depth)?
|  |  |     +--:(deep)
|  |  |     |  +---w deep?        empty
|  |  |     +--:(shallow)
|  |  |        +---w shallow?     empty
|  |  +--ro output
|  |     +--ro (outformat)?
|  |        +--:(cli)
|  |        |  +--ro cli?          string
|  |        +--:(result-xml)
|  |           +--ro result-xml
|  x--ro device-list*                string
|  +---x un-deploy
|  |  +---w input
|  |  |  +---w no-revision-drop?       empty
|  |  |  +---w no-overwrite?           empty
|  |  |  +---w no-networking?          empty
|  |  |  +---w no-out-of-sync-check?   empty
|  |  |  +---w commit-queue!
|  |  |  |  +---w tag?            string
|  |  |  |  +---w (operation-mode)?
|  |  |  |  |  +--:(async)
|  |  |  |  |  |  +---w async?          empty
|  |  |  |  |  +--:(sync)
|  |  |  |  |  |  +---w sync!
|  |  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |  |        +--:(timeout)
|  |  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |  |        +--:(infinity)
|  |  |  |  |  |           +---w infinity?   empty
|  |  |  |  |  +--:(bypass)
|  |  |  |  |     +---w bypass?         empty
|  |  |  |  +---w block-others?   empty
|  |  |  |  +---w lock?           empty
|  |  |  +---w ignore-refcount?        empty
|  |  +--ro output
|  |     +--ro commit-queue
|  |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|  |        +--ro tag?      string
|  |        +--ro status?   enumeration
|  +--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
|  +--ro commit-queue
|  |  +--ro status?                   enumeration
```

```
|  |  +--ro cleared-by-admin?          empty
|  |  +--ro no-of-impacted-services?   uint32
|  |  +--ro failed-device* [name]
|  |  |  +--ro name           -> /ncs:devices/device/name
|  |  |  +--ro time?          yang:date-and-time
|  |  |  +--ro config-data?   string
|  |  |  +--ro error?         string
|  |  +--ro queue-item?                uint64
|  |  +---x clear
|  |  +---x purge
|  +--rw private
|  |  +--rw diff-set?               binary
|  |  +--rw forward-diff-set?       binary
|  |  +--rw device-list*            string
|  |  +--rw service-list*           instance-identifier
|  |  +--rw property-list
|  |  |  +--rw property* [name]
|  |  |     +--rw name     string
|  |  |     +--rw value?   string
|  |  +--rw re-deploy-counter?      int32
|  |  +--rw latest-commit-params?   binary
|  |  +--rw latest-u-info?          binary
|  +--rw name                        string
|  +--rw vnfd                        -> /nfvo:nfvo/vnfd/id
|  +--rw vdu                         -> deref(../vnfd)/../nfvo:vdu/id
|  +--rw esc-device* [name]
|     +--rw name    -> /ncs:devices/device/name
+--rw image* [name]
   +---x check-sync
   |  +---w input
   |  |  +---w outformat?                 outformat4
   |  |  +---w (depth)?
   |  |  |  +--:(deep)
   |  |  |  |  +---w deep?                      empty
   |  |  |  +--:(shallow)
   |  |  |     +---w shallow?                   empty
   |  |  +---w suppress-positive-result?   empty
   |  +--ro output
   |     +--ro (outformat)?
   |        +--:(in-sync)
   |        |  +--ro in-sync?      boolean
   |        +--:(result-xml)
   |        |  +--ro result-xml
   |        +--:(cli)
   |        |  +--ro cli?          string
   |        +--:(native)
   |           +--ro native
   |              +--ro device*
   |                 +--ro name?   string
   |                 +--ro data?   string
   +---x deep-check-sync
   |  +---w input
   |  |  +---w outformat?                 outformat-deep-check-sync
   |  |  +---w suppress-positive-result?   empty
   |  +--ro output
   |     +--ro (outformat)?
   |        +--:(in-sync)
   |        |  +--ro in-sync?      boolean
   |        +--:(result-xml)
   |        |  +--ro result-xml
   |        +--:(cli)
   |           +--ro cli?          string
```

```
+---x re-deploy
|  +---w input
|  |  +---w dry-run!
|  |  |  +---w outformat?   outformat3
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
|  |  +---w (depth)?
|  |  |  +--:(deep)
|  |  |  |  +---w deep?                  empty
|  |  |  +--:(shallow)
|  |  |     +---w shallow?               empty
|  |  +---w reconcile!
|  |     +---w (c-non-service-config)?
|  |        +--:(keep-non-service-config)
|  |        |  +---w keep-non-service-config?      empty
|  |        +--:(discard-non-service-config)
|  |           +---w discard-non-service-config?   empty
|  +--ro output
|     +--ro (outformat)?
|     |  +--:(result-xml)
|     |  |  +--ro result-xml
|     |  +--:(cli)
|     |  |  +--ro cli?            string
|     |  +--:(native)
|     |     +--ro native
|     |        +--ro device*
|     |           +--ro name?   string
|     |           +--ro data?   string
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x reactive-re-deploy
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x touch
x--ro device-modifications?       string
+--ro modified
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+--ro directly-modified
```

```
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+---x get-modifications
|  +---w input
|  |  +---w outformat?   outformat2
|  |  +---w reverse?     empty
|  |  +---w (depth)?
|  |     +--:(deep)
|  |     |  +---w deep?        empty
|  |     +--:(shallow)
|  |        +---w shallow?     empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(cli)
|        |  +--ro cli?          string
|        +--:(result-xml)
|           +--ro result-xml
x--ro device-list*                string
+---x un-deploy
|  +---w input
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
|  |  +---w ignore-refcount?        empty
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
+--ro commit-queue
|  +--ro status?                    enumeration
|  +--ro cleared-by-admin?          empty
|  +--ro no-of-impacted-services?   uint32
|  +--ro failed-device* [name]
|  |  +--ro name           -> /ncs:devices/device/name
|  |  +--ro time?          yang:date-and-time
|  |  +--ro config-data?   string
|  |  +--ro error?         string
|  +--ro queue-item?                uint64
|  +---x clear
|  +---x purge
+--rw private
|  +--rw diff-set?              binary
|  +--rw forward-diff-set?      binary
```

```
          |  +--rw device-list*           string
          |  +--rw service-list*          instance-identifier
          |  +--rw property-list
          |  |  +--rw property* [name]
          |  |     +--rw name     string
          |  |     +--rw value?   string
          |  +--rw re-deploy-counter?      int32
          |  +--rw latest-commit-params?   binary
          |  +--rw latest-u-info?          binary
          +--rw name                        string
          +--rw vnfd                        -> /nfvo:nfvo/vnfd/id
          +--rw vdu                         -> deref(../vnfd)/../nfvo:vdu/id
          +--rw esc-device* [name]
             +--rw name    -> /ncs:devices/device/name
augment /nfvo:nfvo:
   +--rw tailf-esc-internal
   |  +--rw vnf-deployment* [tenant deployment-name]
   |     +---x check-sync
   |     |  +---w input
   |     |  |  +---w outformat?                 outformat4
   |     |  |  +---w (depth)?
   |     |  |  |  +--:(deep)
   |     |  |  |  |  +---w deep?                     empty
   |     |  |  |  +--:(shallow)
   |     |  |  |     +---w shallow?                  empty
   |     |  |  +---w suppress-positive-result?   empty
   |     |  +--ro output
   |     |     +--ro (outformat)?
   |     |        +--:(in-sync)
   |     |        |  +--ro in-sync?      boolean
   |     |        +--:(result-xml)
   |     |        |  +--ro result-xml
   |     |        +--:(cli)
   |     |        |  +--ro cli?          string
   |     |        +--:(native)
   |     |           +--ro native
   |     |              +--ro device*
   |     |                 +--ro name?   string
   |     |                 +--ro data?   string
   |     +---x deep-check-sync
   |     |  +---w input
   |     |  |  +---w outformat?                 outformat-deep-check-sync
   |     |  |  +---w suppress-positive-result?   empty
   |     |  +--ro output
   |     |     +--ro (outformat)?
   |     |        +--:(in-sync)
   |     |        |  +--ro in-sync?      boolean
   |     |        +--:(result-xml)
   |     |        |  +--ro result-xml
   |     |        +--:(cli)
   |     |           +--ro cli?          string
   |     +---x re-deploy
   |     |  +---w input
   |     |  |  +---w dry-run!
   |     |  |  |  +---w outformat?   outformat3
   |     |  |  +---w no-revision-drop?       empty
   |     |  |  +---w no-overwrite?           empty
   |     |  |  +---w no-networking?          empty
   |     |  |  +---w no-out-of-sync-check?   empty
   |     |  |  +---w commit-queue!
   |     |  |  |  +---w tag?            string
   |     |  |  |  +---w (operation-mode)?
```

```
|      |  |  |  |  +--:(async)
|      |  |  |  |  |  +---w async?          empty
|      |  |  |  |  +--:(sync)
|      |  |  |  |  |  +---w sync!
|      |  |  |  |  |     +---w (time-out-choice)?
|      |  |  |  |  |        +--:(timeout)
|      |  |  |  |  |        |  +---w timeout?    uint32
|      |  |  |  |  |        +--:(infinity)
|      |  |  |  |  |           +---w infinity?   empty
|      |  |  |  |  +--:(bypass)
|      |  |  |  |     +---w bypass?         empty
|      |  |  |  +---w block-others?   empty
|      |  |  |  +---w lock?           empty
|      |  |  +---w (depth)?
|      |  |  |  +--:(deep)
|      |  |  |  |  +---w deep?                  empty
|      |  |  |  +--:(shallow)
|      |  |  |     +---w shallow?               empty
|      |  |  +---w reconcile!
|      |  |     +---w (c-non-service-config)?
|      |  |        +--:(keep-non-service-config)
|      |  |        |  +---w keep-non-service-config?      empty
|      |  |        +--:(discard-non-service-config)
|      |  |           +---w discard-non-service-config?   empty
|      |  +--ro output
|      |     +--ro (outformat)?
|      |     |  +--:(result-xml)
|      |     |  |  +--ro result-xml
|      |     |  +--:(cli)
|      |     |  |  +--ro cli?            string
|      |     |  +--:(native)
|      |     |     +--ro native
|      |     |        +--ro device*
|      |     |           +--ro name?   string
|      |     |           +--ro data?   string
|      |     +--ro commit-queue
|      |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|      |        +--ro tag?      string
|      |        +--ro status?   enumeration
|      +---x reactive-re-deploy
|      |  +--ro output
|      |     +--ro commit-queue
|      |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|      |        +--ro tag?      string
|      |        +--ro status?   enumeration
|      +---x touch
|      x--ro device-modifications?       string
|      +--ro modified
|      |  +--ro devices*    -> /ncs:devices/device/name
|      |  +--ro services*   instance-identifier
|      +--ro directly-modified
|      |  +--ro devices*    -> /ncs:devices/device/name
|      |  +--ro services*   instance-identifier
|      +---x get-modifications
|      |  +---w input
|      |  |  +---w outformat?   outformat2
|      |  |  +---w reverse?     empty
|      |  |  +---w (depth)?
|      |  |     +--:(deep)
|      |  |     |  +---w deep?        empty
|      |  |     +--:(shallow)
|      |  |        +---w shallow?     empty
```

```
|     |  +--ro output
|     |     +--ro (outformat)?
|     |        +--:(cli)
|     |        |  +--ro cli?          string
|     |        +--:(result-xml)
|     |           +--ro result-xml
|     x--ro device-list*                string
|     +---x un-deploy
|     |  +---w input
|     |  |  +---w no-revision-drop?       empty
|     |  |  +---w no-overwrite?           empty
|     |  |  +---w no-networking?          empty
|     |  |  +---w no-out-of-sync-check?   empty
|     |  |  +---w commit-queue!
|     |  |  |  +---w tag?            string
|     |  |  |  +---w (operation-mode)?
|     |  |  |  |  +--:(async)
|     |  |  |  |  |  +---w async?          empty
|     |  |  |  |  +--:(sync)
|     |  |  |  |  |  +---w sync!
|     |  |  |  |  |     +---w (time-out-choice)?
|     |  |  |  |  |        +--:(timeout)
|     |  |  |  |  |        |  +---w timeout?    uint32
|     |  |  |  |  |        +--:(infinity)
|     |  |  |  |  |           +---w infinity?   empty
|     |  |  |  |  +--:(bypass)
|     |  |  |  |     +---w bypass?         empty
|     |  |  |  +---w block-others?   empty
|     |  |  |  +---w lock?           empty
|     |  |  +---w ignore-refcount?        empty
|     |  +--ro output
|     |     +--ro commit-queue
|     |        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|     |        +--ro tag?      string
|     |        +--ro status?   enumeration
|     +--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
|     +--ro commit-queue
|     |  +--ro status?                    enumeration
|     |  +--ro cleared-by-admin?          empty
|     |  +--ro no-of-impacted-services?   uint32
|     |  +--ro failed-device* [name]
|     |  |  +--ro name           -> /ncs:devices/device/name
|     |  |  +--ro time?          yang:date-and-time
|     |  |  +--ro config-data?   string
|     |  |  +--ro error?         string
|     |  +--ro queue-item?                uint64
|     |  +---x clear
|     |  +---x purge
|     +--rw private
|     |  +--rw diff-set?               binary
|     |  +--rw forward-diff-set?       binary
|     |  +--rw device-list*            string
|     |  +--rw service-list*           instance-identifier
|     |  +--rw property-list
|     |  |  +--rw property* [name]
|     |  |     +--rw name     string
|     |  |     +--rw value?   string
|     |  +--rw re-deploy-counter?      int32
|     |  +--rw latest-commit-params?   binary
|     |  +--rw latest-u-info?          binary
|     +--rw tenant                      -> /ncs:devices/device/config/esc:esc_datamodel/tenan
|     +--rw deployment-name             string
```

```
|     +--rw username                    string
|     +--rw esc-device* [name]
|     |  +--rw name    -> /ncs:devices/device/name
|     +--rw staged-delete?              empty
|     +--rw vnfr* [id]
|     |  +--rw id                       -> /nfvo:nfvo/vnfd/id
|     |  +--rw vnfd-flavor              -> deref(../id)/../nfvo:deployment-flavor/id
|     |  +--rw instantiation-level      -> deref(../vnfd-flavor)/../nfvo:instantiation-lev
|     |  +--rw managed?                 empty
|     |  +--rw vdu* [id]
|     |  |  +--rw id                          -> deref(../../id)/../nfvo:vdu/id
|     |  |  +--rw image-name                  string
|     |  |  +--rw flavor-name                 string
|     |  |  +--rw day0* [destination]
|     |  |  |  +--rw destination    string
|     |  |  |  +--rw (source)?
|     |  |  |  |  +--:(url)
|     |  |  |  |  |  +--rw url?           inet:uri
|     |  |  |  |  +--:(data)
|     |  |  |  |     +--rw data?          string
|     |  |  |  +--rw variable* [name]
|     |  |  |     +--rw name     string
|     |  |  |     +--rw value?   string
|     |  |  +--rw authgroup?                  -> /ncs:devices/authgroups/group/name
|     |  |  +--rw connection-point-address* [id]
|     |  |  |  +--rw id         -> /nfvo:nfvo/vnfd[nfvo:id=current()/../../../id]/vdu[nfvo
|     |  |  |  +--rw address?   inet:ip-address
|     |  |  |  +--rw start      inet:ip-address
|     |  |  |  +--rw end        inet:ip-address
|     |  |  +--rw device-template* [name]
|     |  |     +--rw name        -> /ncs:devices/template/name
|     |  |     +--rw variable* [name]
|     |  |        +--rw name     string
|     |  |        +--rw value?   string
|     |  +--rw virtual-link* [id]
|     |  |  +--rw id        -> deref(../../id)/../nfvo:virtual-link/id
|     |  |  +--rw dhcp?     empty
|     |  |  +--rw subnet
|     |  |     +--rw network       inet:ip-prefix
|     |  |     +--rw (gateway-choice)?
|     |  |        +--:(gateway)
|     |  |        |  +--rw gateway?      inet:ip-address
|     |  |        +--:(no-gateway)
|     |  |           +--rw no-gateway?   empty
|     |  +--rw vnfd-connection-point* [id]
|     |     +--rw id     -> deref(../../id)/../nfvo:external-connection-point/id
|     |     +--rw vlr    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
|     +--rw vlr* [id]
|        +--rw id    -> /nfvo:nfvo/vlr/nfvo-esc:esc/vlr/id
+--rw settings-esc!
   +---x check-sync
   |  +---w input
   |  |  +---w outformat?                 outformat4
   |  |  +---w (depth)?
   |  |  |  +--:(deep)
   |  |  |  |  +---w deep?                      empty
   |  |  |  +--:(shallow)
   |  |  |     +---w shallow?                   empty
   |  |  +---w suppress-positive-result?   empty
   |  +--ro output
   |     +--ro (outformat)?
   |        +--:(in-sync)
```

```
|        |  +--ro in-sync?      boolean
|        +--:(result-xml)
|        |  +--ro result-xml
|        +--:(cli)
|        |  +--ro cli?          string
|        +--:(native)
|           +--ro native
|              +--ro device*
|                 +--ro name?   string
|                 +--ro data?   string
+---x deep-check-sync
|  +---w input
|  |  +---w outformat?                  outformat-deep-check-sync
|  |  +---w suppress-positive-result?   empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(in-sync)
|        |  +--ro in-sync?      boolean
|        +--:(result-xml)
|        |  +--ro result-xml
|        +--:(cli)
|           +--ro cli?          string
+---x re-deploy
|  +---w input
|  |  +---w dry-run!
|  |  |  +---w outformat?   outformat3
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
|  |  +---w (depth)?
|  |  |  +--:(deep)
|  |  |  |  +---w deep?                   empty
|  |  |  +--:(shallow)
|  |  |     +---w shallow?                empty
|  |  +---w reconcile!
|  |     +---w (c-non-service-config)?
|  |        +--:(keep-non-service-config)
|  |        |  +---w keep-non-service-config?      empty
|  |        +--:(discard-non-service-config)
|  |           +---w discard-non-service-config?   empty
|  +--ro output
|     +--ro (outformat)?
|     |  +--:(result-xml)
|     |  |  +--ro result-xml
|     |  +--:(cli)
```

```
|     |  |  +--ro cli?            string
|     |  +--:(native)
|     |     +--ro native
|     |        +--ro device*
|     |           +--ro name?   string
|     |           +--ro data?   string
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x reactive-re-deploy
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+---x touch
x--ro device-modifications?       string
+--ro modified
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+--ro directly-modified
|  +--ro devices*    -> /ncs:devices/device/name
|  +--ro services*   instance-identifier
+---x get-modifications
|  +---w input
|  |  +---w outformat?   outformat2
|  |  +---w reverse?     empty
|  |  +---w (depth)?
|  |     +--:(deep)
|  |     |  +---w deep?        empty
|  |     +--:(shallow)
|  |        +---w shallow?     empty
|  +--ro output
|     +--ro (outformat)?
|        +--:(cli)
|        |  +--ro cli?          string
|        +--:(result-xml)
|           +--ro result-xml
x--ro device-list*                string
+---x un-deploy
|  +---w input
|  |  +---w no-revision-drop?       empty
|  |  +---w no-overwrite?           empty
|  |  +---w no-networking?          empty
|  |  +---w no-out-of-sync-check?   empty
|  |  +---w commit-queue!
|  |  |  +---w tag?            string
|  |  |  +---w (operation-mode)?
|  |  |  |  +--:(async)
|  |  |  |  |  +---w async?          empty
|  |  |  |  +--:(sync)
|  |  |  |  |  +---w sync!
|  |  |  |  |     +---w (time-out-choice)?
|  |  |  |  |        +--:(timeout)
|  |  |  |  |        |  +---w timeout?    uint32
|  |  |  |  |        +--:(infinity)
|  |  |  |  |           +---w infinity?   empty
|  |  |  |  +--:(bypass)
|  |  |  |     +---w bypass?         empty
|  |  |  +---w block-others?   empty
|  |  |  +---w lock?           empty
```

```
|  |  +---w ignore-refcount?        empty
|  +--ro output
|     +--ro commit-queue
|        +--ro id?       -> /ncs:devices/commit-queue/queue-item/id
|        +--ro tag?      string
|        +--ro status?   enumeration
+--ro used-by-customer-service*   -> /ncs:services/customer-service/object-id
+--ro commit-queue
|  +--ro status?                    enumeration
|  +--ro cleared-by-admin?          empty
|  +--ro no-of-impacted-services?   uint32
|  +--ro failed-device* [name]
|  |  +--ro name           -> /ncs:devices/device/name
|  |  +--ro time?          yang:date-and-time
|  |  +--ro config-data?   string
|  |  +--ro error?         string
|  +--ro queue-item?                uint64
|  +---x clear
|  +---x purge
+--rw private
|  +--rw diff-set?              binary
|  +--rw forward-diff-set?      binary
|  +--rw device-list*           string
|  +--rw service-list*          instance-identifier
|  +--rw property-list
|  |  +--rw property* [name]
|  |     +--rw name     string
|  |     +--rw value?   string
|  +--rw re-deploy-counter?     int32
|  +--rw latest-commit-params?  binary
|  +--rw latest-u-info?         binary
+--rw netconf-subscription
   +--rw username      string
   +--rw esc-device* [name]
      +--rw name    -> /ncs:devices/device/name
```