blueplanet®
a division of Ciena

Blue Planet NFVO:
Network Service Descriptor Development

C861, Revision B

Agenda

NFVO Components and Architecture

NFVO Operations

- Setup Domains and Devices
- Service Instantiation
- VNF Scaling

Creating NFVO Network Service Designs

- Flavors and Levels
- Service Chaining
- UI Design Tools
- Onboarding VNFD and NSD files
- Create NFVO Descriptors – NSD and VNFD

Lifecycle Operations

- NSD Version Updates
- VNF Software Updates

NSD and VNFD Package Import and Export

NFVO Components and Architecture

NFVO: How It Works

VNF and Network Service Descriptors (VNFDs, NSDs) streamline VNF on-boarding and facilitate the deployment of network services comprised of multiple VNFs

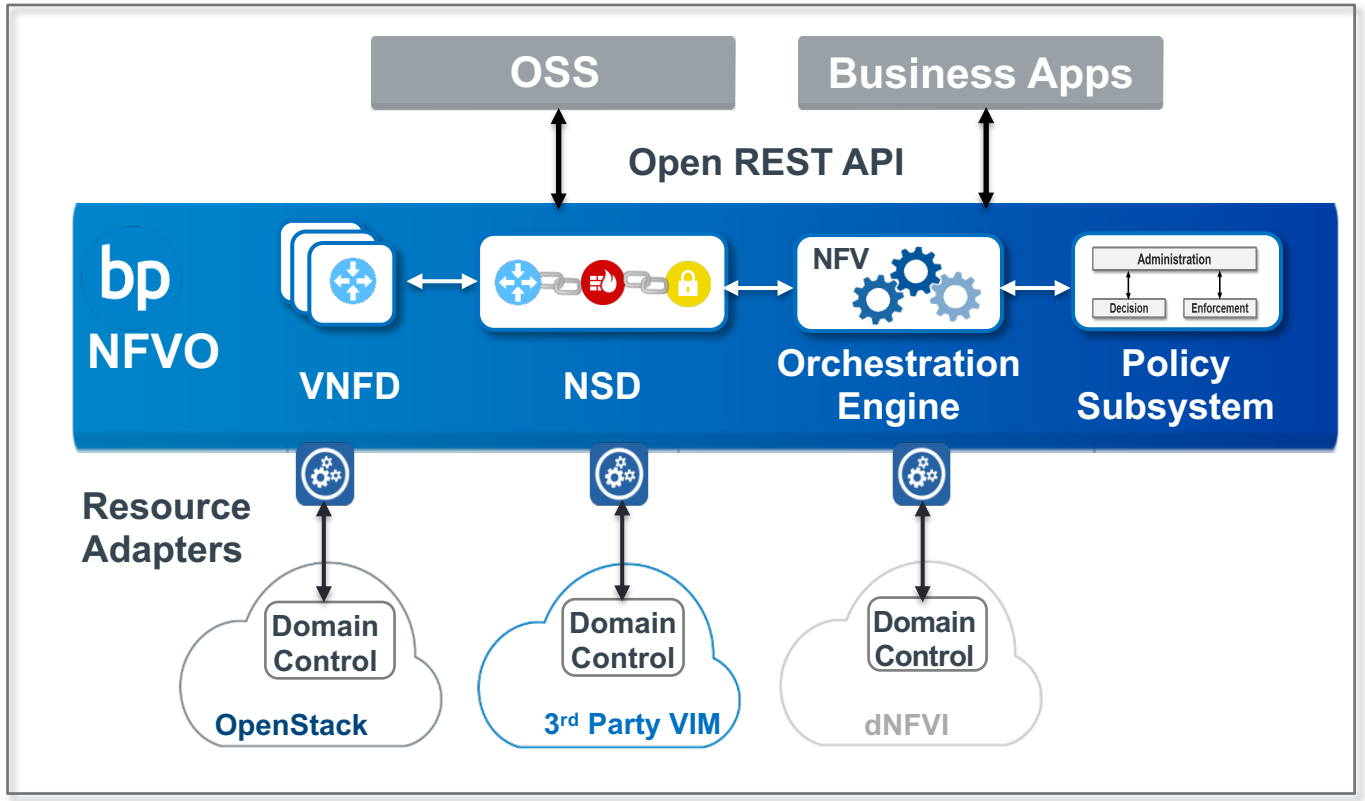
NFV Orchestration Engine automates end-to-end service creation and lifecycle management

Resource Adapters (RAs) extend BP control to underlying VIMs using native protocols

Open REST API simplifies integration with OSS and business applications

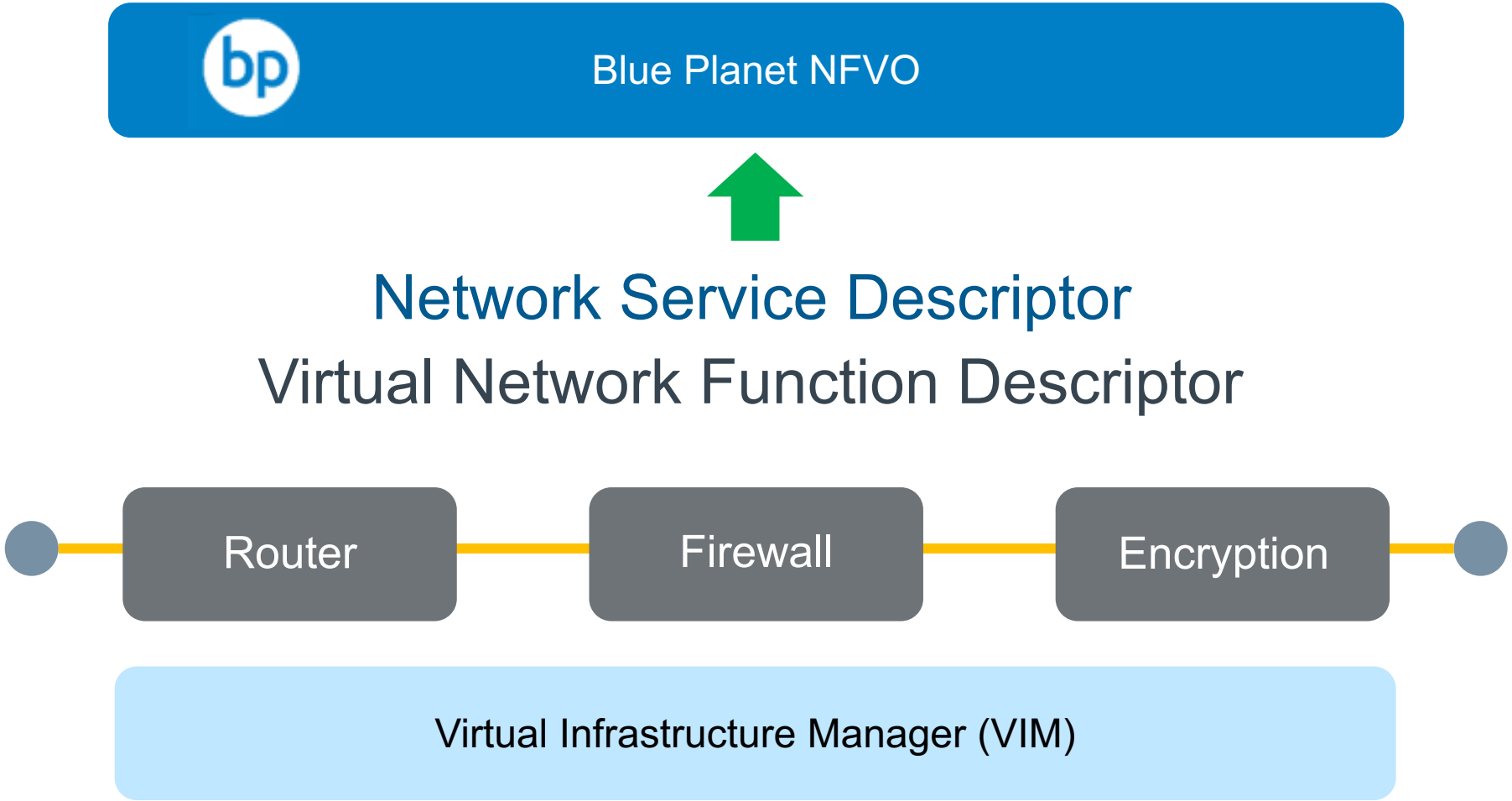
Policy Subsystem enables the flexible creation and enforcement of rules to control network resources

Microservices-based platform ensures smooth migration to cloud-native networks, with optimal agility, scale and performance



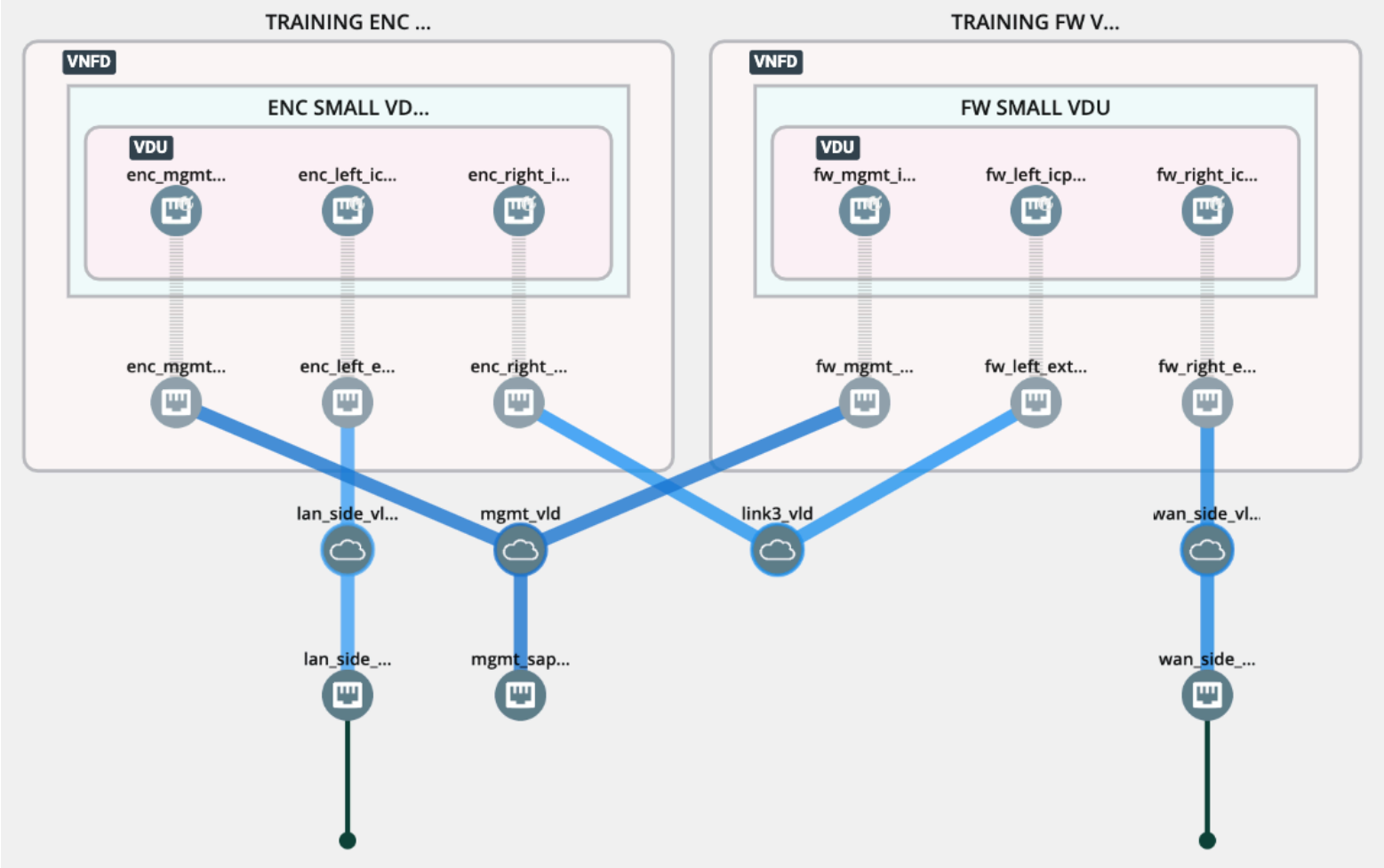
Blue Planet NFVO Architecture and Components

blueplanet
a division of Ciena



What is a Network Service?

- **A service chain of one or more VNFs instantiated in a virtual infrastructure manager**



Network Functions Virtualisation (NFV); Management and Orchestration

References

ETSI:

- ETSI GS NFV-IFA 011
  - Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification
- ETSI GS NFV-IFA 014
  - Network Functions Virtualisation (NFV); Management and Orchestration; Network Service Templates Specification
- ETSI GS NFV 002
  - Network Function Virtualisation (NFV); Architectural Framework
- ESTI GS NFV-SOL 003
  - Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point
- ETSI NFV-IFA 005
  - Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification
- ETSI NFV-IFA 006
  - Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification

Ciena

- Blue Planet NFVO Developer Guide

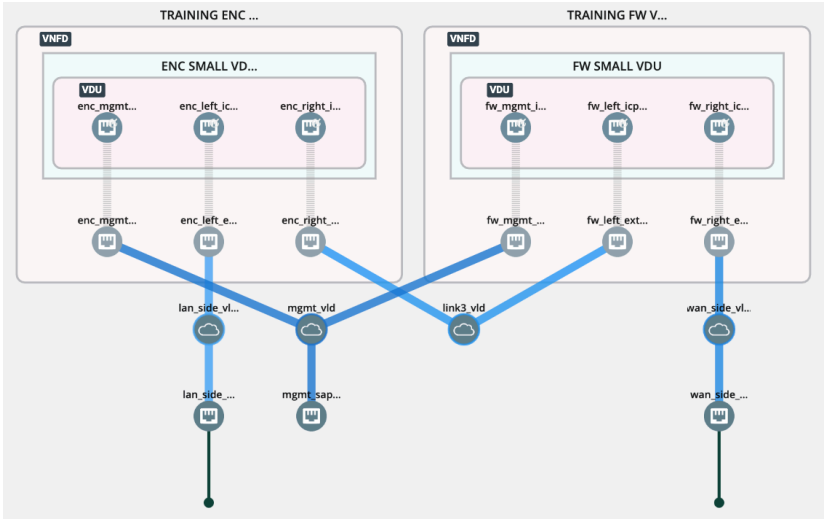
Descriptors

PNF Descriptor

VNF Descriptor

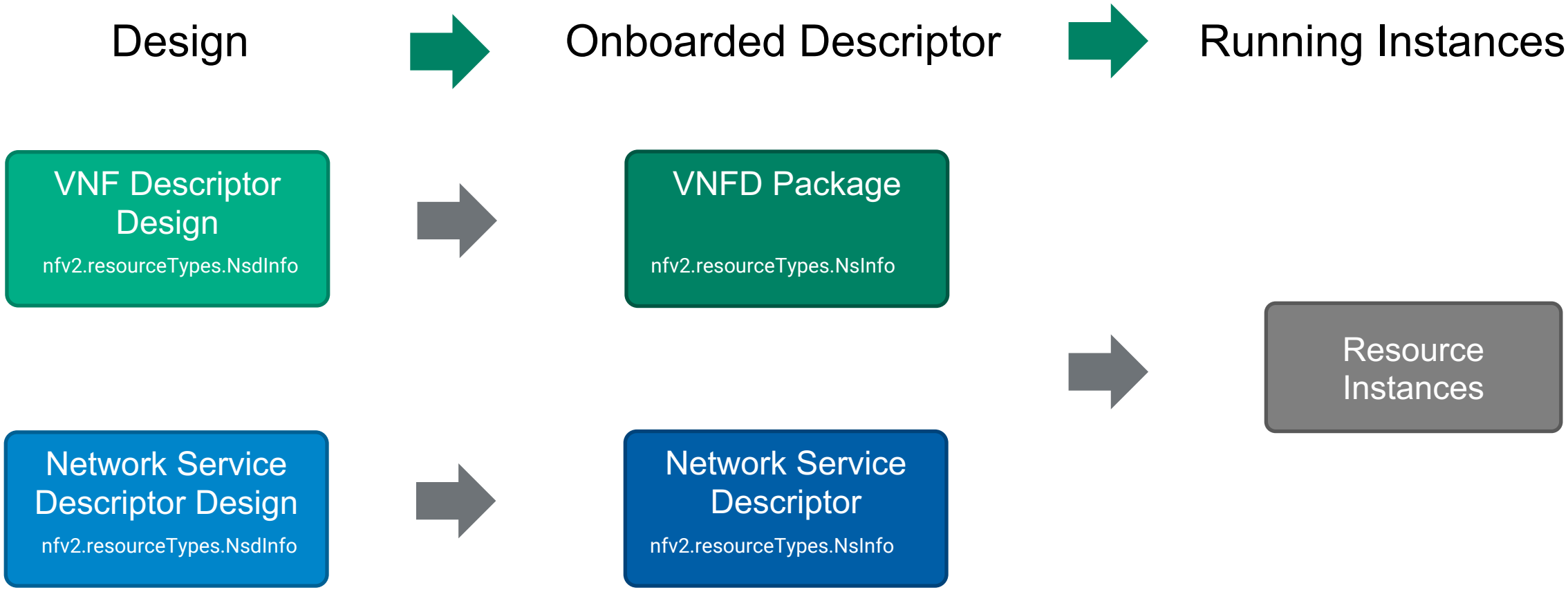
NS Descriptor

Running Instances



VNFs running in a VIM with required virtual networking

NFVO Catalogs



Catalogs

NS Designs that have onboarded and can be used for instantiation. VNFDs must be onboarded.

Tool for creating NS designs

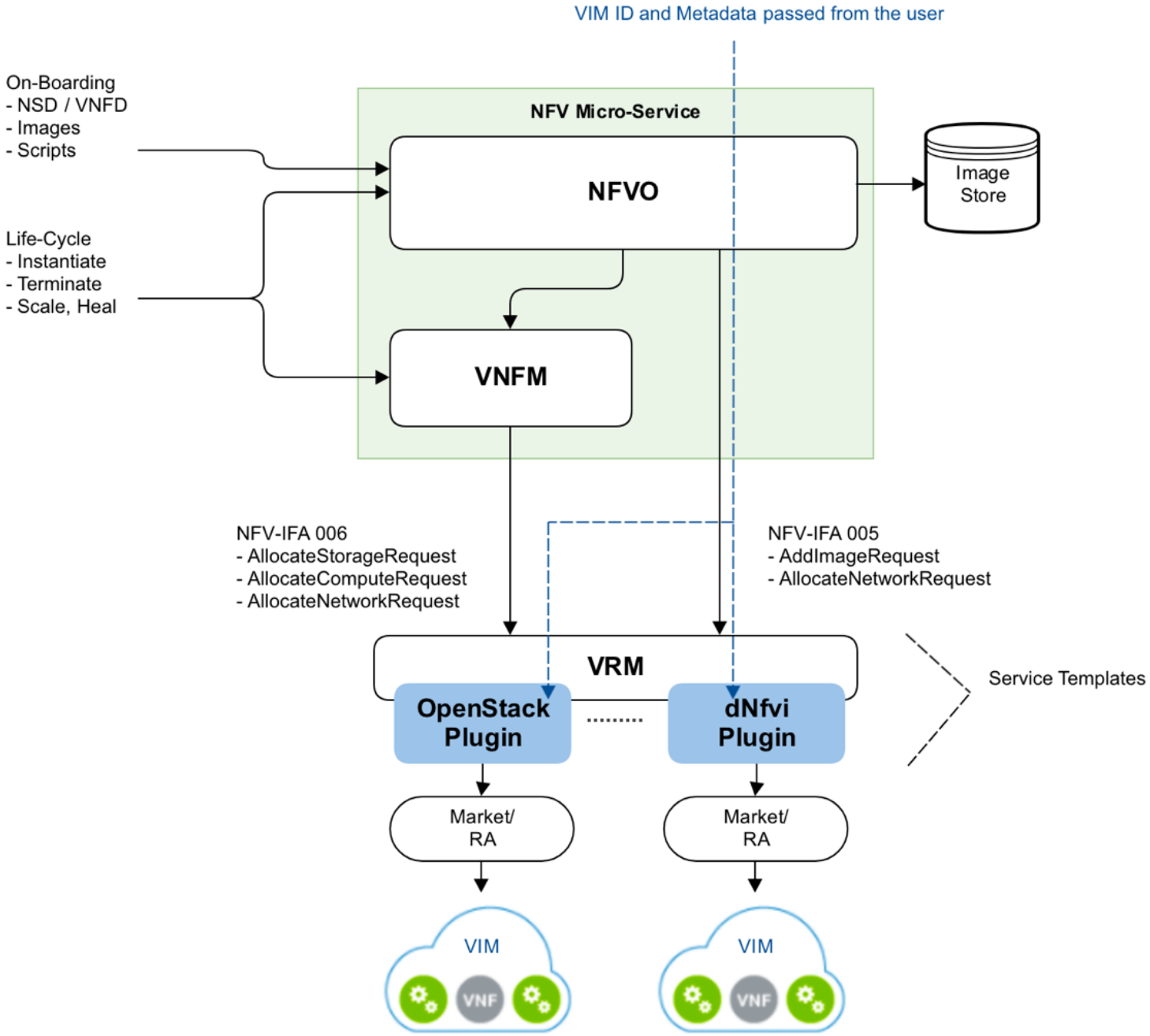
NS Designs that have been imported/created in Blue Planet

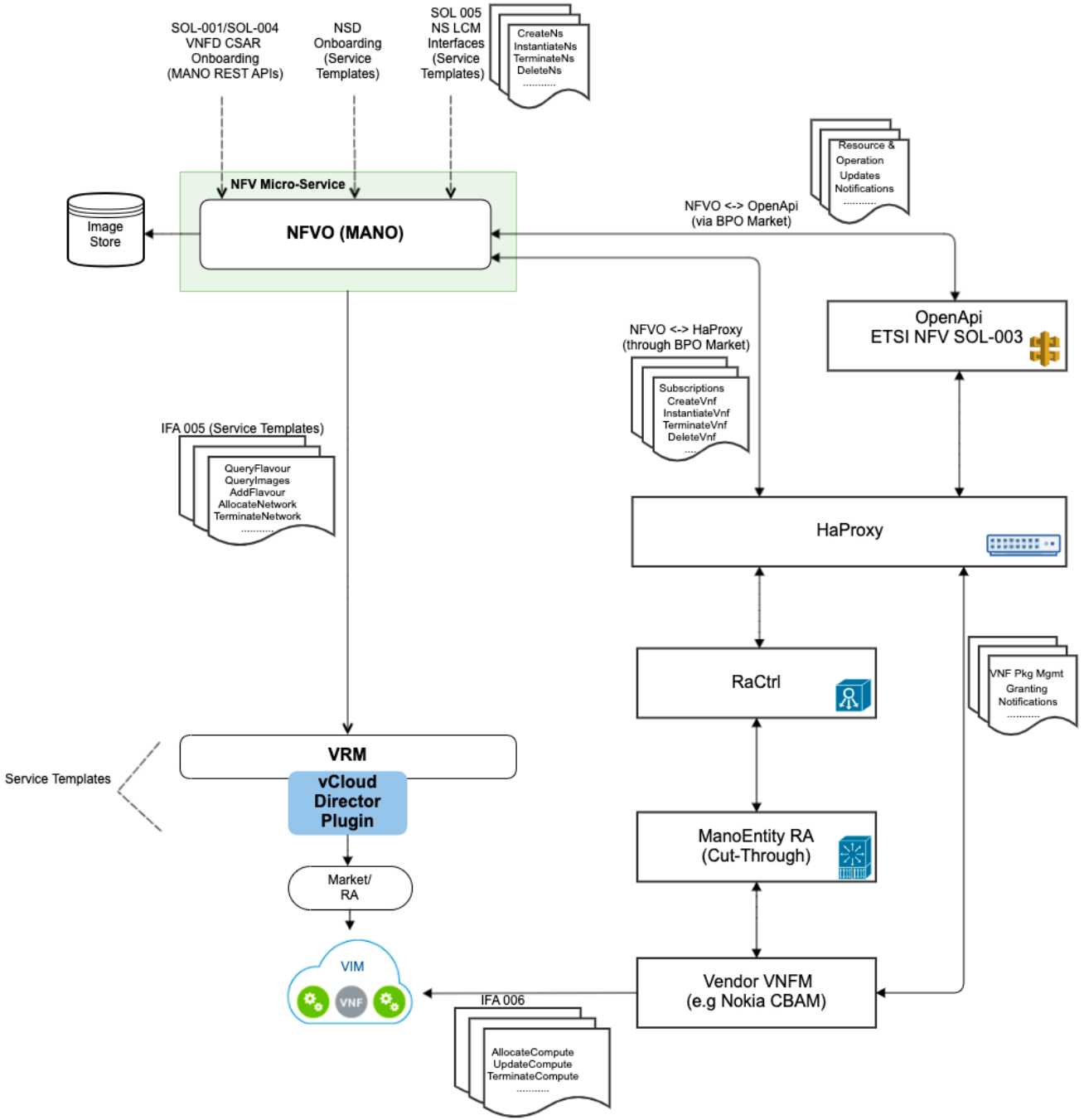
VNF Designs that have been imported/created in Blue Planet

VNF Designs that have onboarded and can be used in onboarded NSDs.

VNF Manager (VNFM)

- **Generic VNFM**
  - Works with NFVO to handle lifecycle operations for VNFs
  - Virtual Resource Manager (VRM) houses plugins for VIMs
  - Plugins adapt domain specific RAs to work with NFVO
    - Plugins are essentially an adaption layer between the RA and NFVO
  - Plugins and RAs can be added independent of NFVO
  - VRM provides the interfaces specified in the ETSI NFV-IFA 005 and ETSI NFV-IFA 006 standards
  - D-NFVI and OpenStack have plugins
- **Specialized VNFM (S-VNFM)**
  - NFVO identifies if the built-in VNFM or a specialized VNFM is being used
    - S-VNFM identified in VNFD
  - Communication to S-VNFM is via the BP ManoEntity RA
  - RA relays SOL 003 compliant REST queries to S-VNFM
  - Each S-VNFM requires a domain
  - Queries initiated by the S-VNFM toward NFVO are served by the SOL 003 OpenApi plugin


VIM ID and Metadata passed from the user
On-Boarding
- NSD / VNFD
- Images
- Scripts
Life-Cycle
- Instantiate
- Terminate
- Scale, Heal
NFV Micro-Service
NFVO
VNFM
Image Store
NFV-IFA 006
- AllocateStorageRequest
- AllocateComputeRequest
- AllocateNetworkRequest
NFV-IFA 005
- AddImageRequest
- AllocateNetworkRequest
VRM
OpenStack Plugin
dNfvi Plugin
Service Templates
Market/ RA
Market/ RA
VIM
VNF
VIM
VNF


SOL-001/SOL-004
VNFD CSAR
Onboarding
(MANO REST APIs)
NSD
Onboarding
(Service
Templates)
SOL 005
NS LCM
Interfaces
(Service
Templates)
CreateNs
InstantiateNs
TerminateNs
DeleteNs
NFV Micro-Service
Image
Store
NFVO (MANO)
Resource &
Operation
Updates
Notifications
NFVO <-> OpenApi
(via BPO Market)
OpenApi
ETSI NFV SOL-003
NFVO <-> HaProxy
(through BPO Market)
Subscriptions
CreateVnf
InstantiateVnf
TerminateVnf
DeleteVnf
IFA 005 (Service Templates)
QueryFlavour
QueryImages
AddFlavour
AllocateNetwork
TerminateNetwork
HaProxy
RaCtrl
VNF Pkg Mgmt
Granting
Notifications
Service Templates
VRM
vCloud
Director
Plugin
Market/
RA
ManoEntity RA
(Cut-Through)
VIM
VNF
IFA 006
Vendor VNFM
(e.g Nokia CBAM)
AllocateCompute
UpdateCompute
TerminateCompute

NFVO Operations

Setting Up NFVO

1. **Install Blue Planet server.**
2. **Deploy RAs using Solution Manager:**
   - D-NFVI
   - OpenStack
3. **Create Domains for each OpenStack and/or D-NFVI.**
   - D-NFVI: Add devices
4. **Create domains for each S-VNFM (as needed).**
5. **Create mock VIM (for development only)**
6. **Setup FTP server.**
7. **Create and onboard VNF Descriptors (VNFD).**
8. **Create and onboard PNF Descriptors.***
9. **Create and onboard Network Service Descriptors (NSD).**
10. **Instantiate and manage network services.**

Lab 1: Setup NFVO

- **Instructor demonstration**
- **Blue Planet NFVO 20.06**
  - Instructor will provide login information

Network Service Instantiation

- **Requires an onboarded Network Service Descriptor (NSD)**
- **Process:**
  - Create NsInfo resource instance
  - Run *Instantiate* operation
  - Enter service details
- **Instantiation wizard simplifies the process:**
  - Creates an NsInfo resource.
  - Identifies the default VNF instantiation levels.
  - Discovers and defines SAPDs.
  - Compiles the network service instantiation operation input data and launches the operation.
- **Recommendation: Use the instantiation wizard**

Metadata

- **Provides a way to specify VIM specific parameters that are not part of the ETSI specification**
- **Data is passed to VIM during instantiation**
- **Metadata can be included in NSD or VNFD**
- **Metadata can be added during instantiation**
  - Will override metadata in descriptors

Lab 2: Instantiate a Network Service

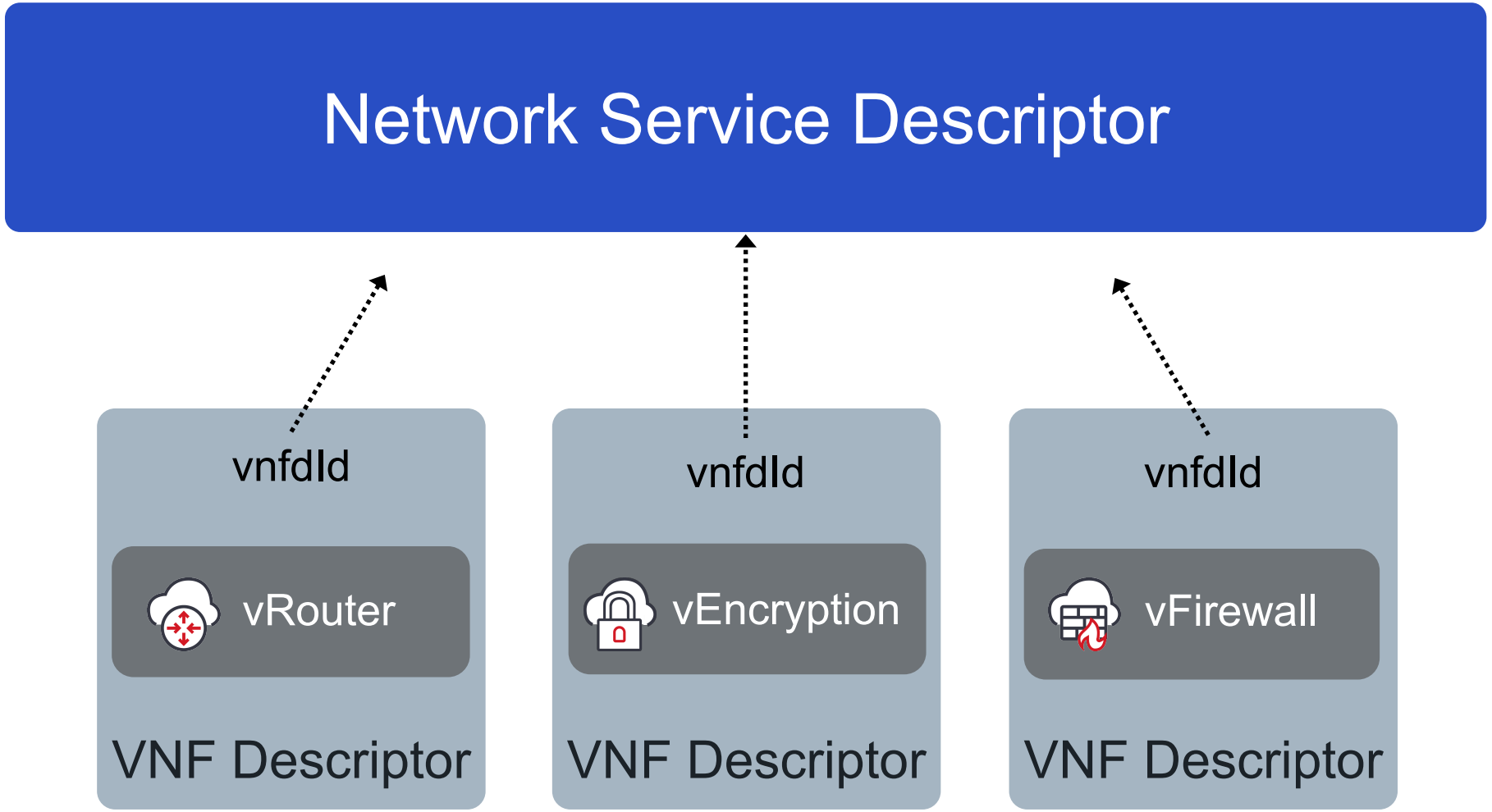
Lab 3: Change NS Instantiation Level

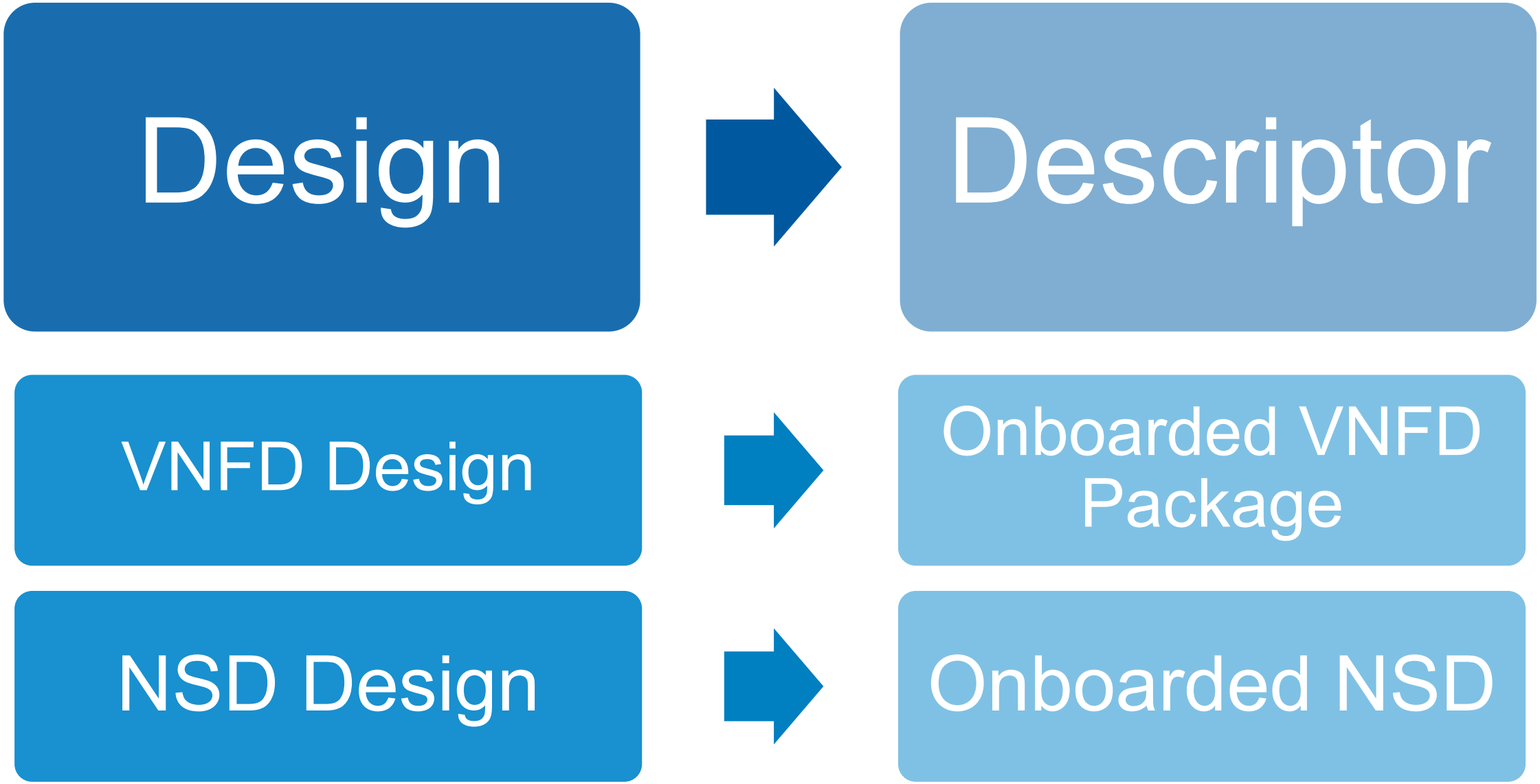
Creating Network Service Designs

Creating Network Service Descriptors

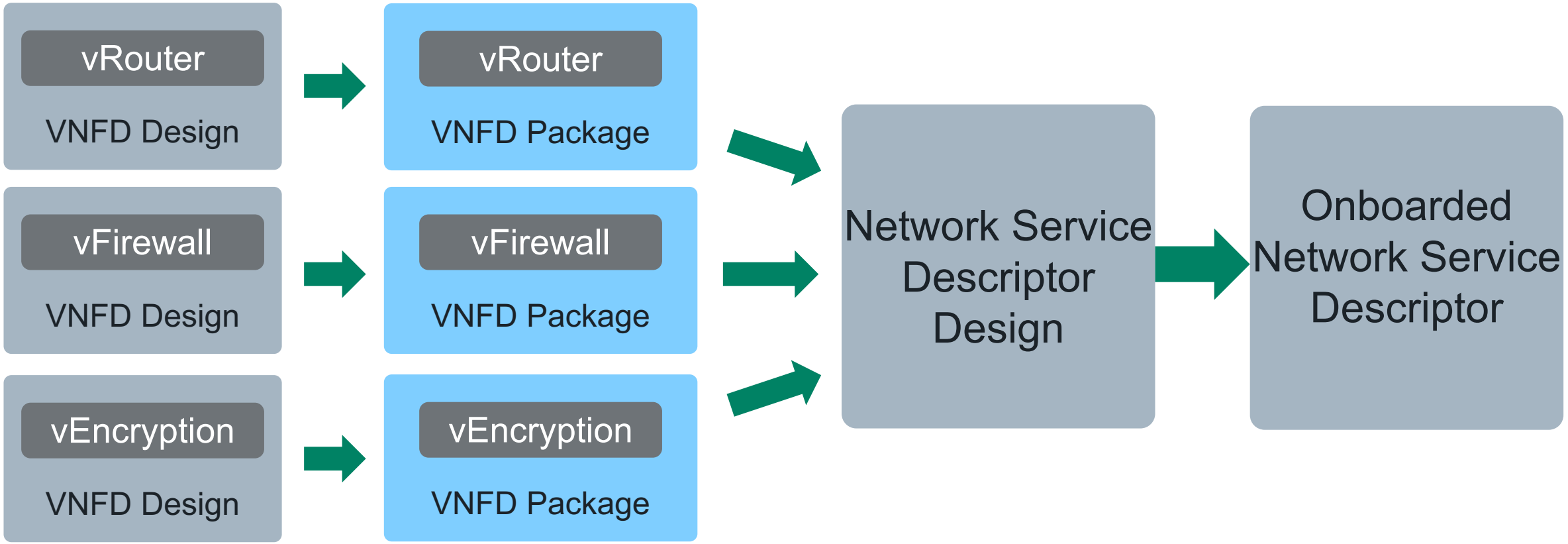
Start with a service design

- Identify VNFs
- Identify virtual interfaces
- Determine traffic flow
- Determine horizontal and vertical scaling requirements
- Understand VIM requirements
- Document VNF compute requirements





blueplanet
a division of Ciena

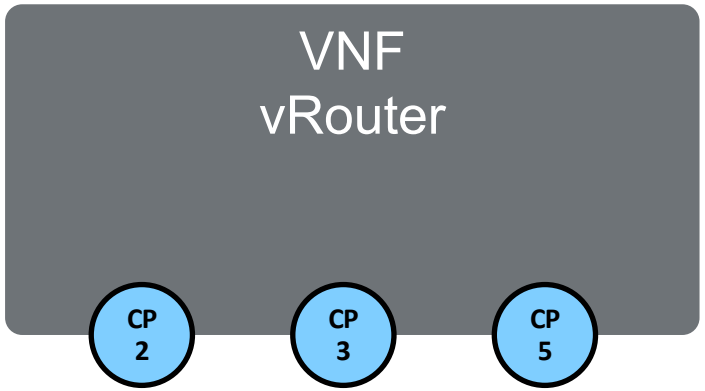


A descriptor can describe multiple possible deployments, one of which is instantiated by a runtime instance.

What is a VNF Descriptor?

A deployment template which describes a VNF in terms of deployment and operational behavior requirements. It also contains connectivity, interface and virtualized resource requirements.

A VNFD Design instance can deploy a VNFD Resource, its related descriptor resources and their relationships.

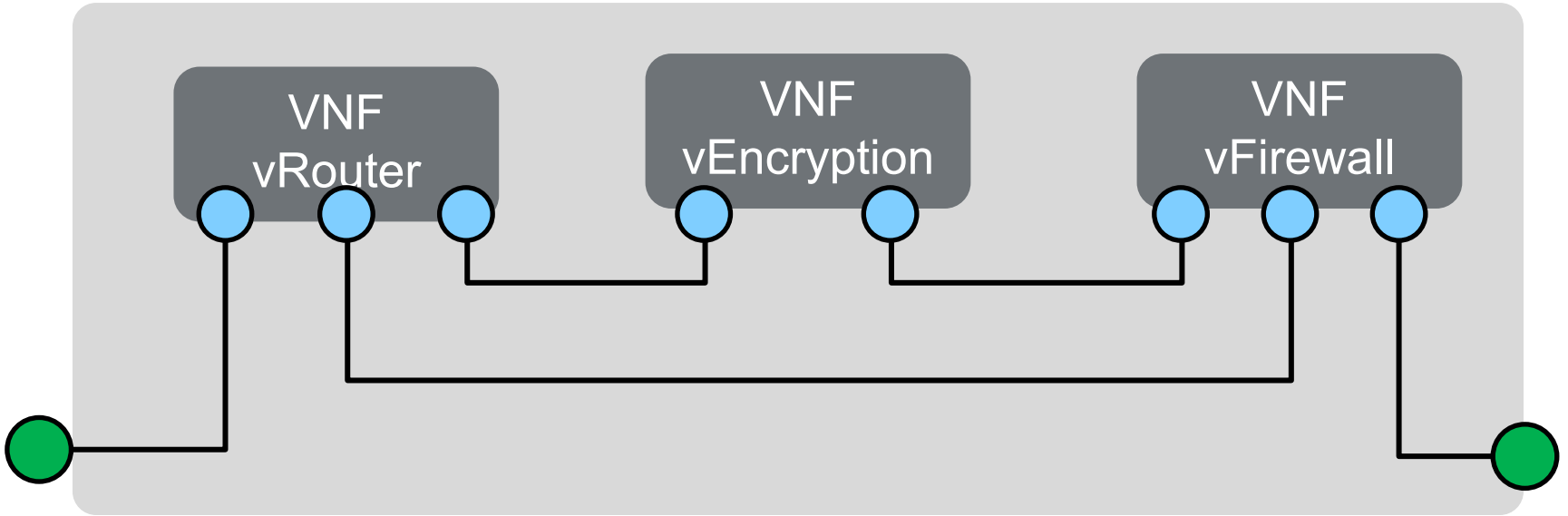


What is a Network Service Descriptor?

Deployment template containing information used by BP NFVO for life cycle management of a network service.

Includes references to the descriptors of its constituent objects:

- Zero, one or more Virtualised Network Function Descriptors (VNFD)
- Zero, one or more Physical Network connect PNFs to VLs
- Zero, one or more nested NSD



Major Model Version

- **V1**
  - First version of VNFD and NSD models
  - Works on pre-20.02 releases
  - Supported in current NFVO release
- **V2**
  - New version to support changes in the ETSI model and SOL003
  - Major changes:
    - NSD:
      - Network forwarding path moved NS deployment flavor
    - VNFD:
      - Instantiation levels moved to deployment flavor
      - Support for specifying VNF manager for SOL 003
  - Only used if you are using SOL003 and an external VNF manager

NFVO Operations

Operations are based on the Network Service Descriptor and VNF Descriptors

- Descriptors must be created to accommodate scaling and upgrades

Operations:

- Service management:
  - Instantiation using wizard
  - Termination and deletion
- Updating a Network Service:
  - Change Flavor
  - Change Level
- NS Operations that require new NSD:
  - Update VNF Software
  - Update NSD version
- VNF Scaling

Consider lifecycle operations that will be needed when you create NS and VNF descriptors

Methods for Creating Designs

Use Blue Planet UI

- VNF Designs
- NS Design

Write and import JSON files

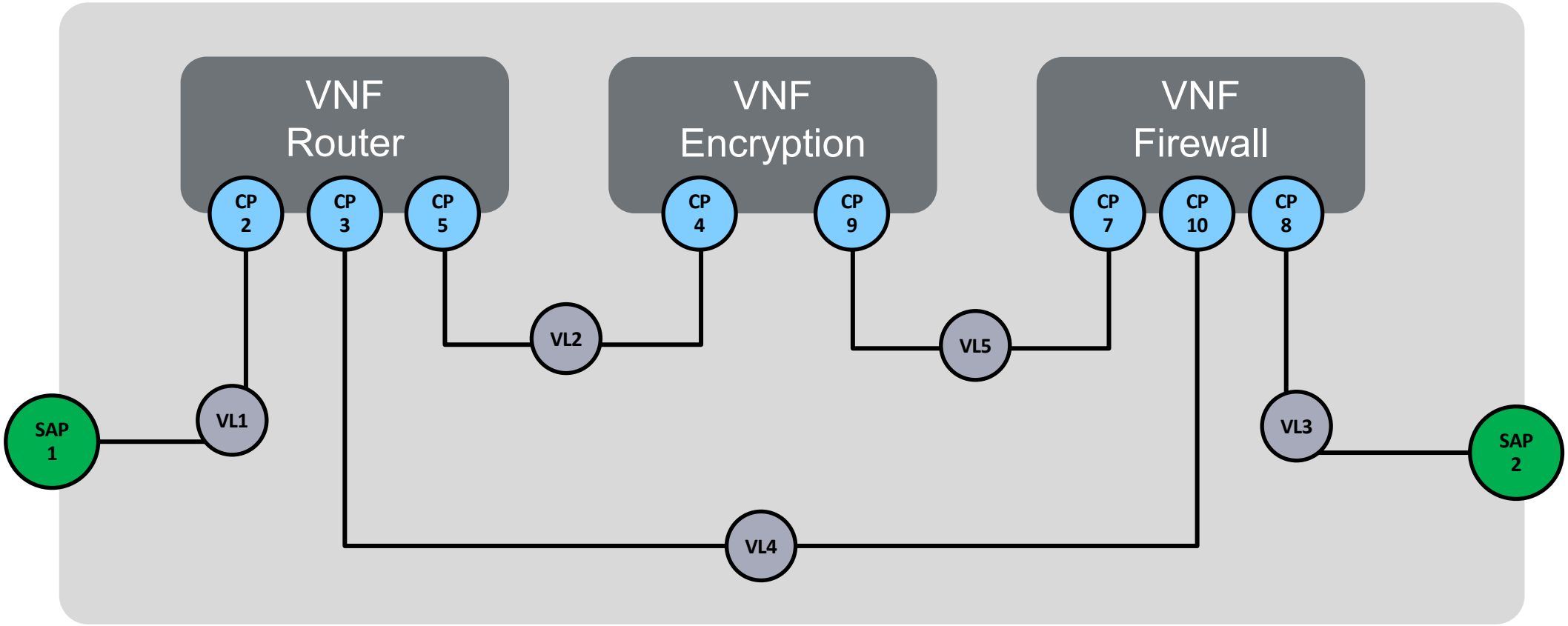
Upload VNF Packages

What you will create…

Network Service Designs

- Design 1: New NSD
  - Two VNFs:
    - Firewall
    - Encryption
  - Two flavors
  - One level
- Design 2: Extension of Design #1
  - Three VNFs:
    - Firewall
    - Encryption
    - Router
  - Three flavors
  - Two forwarding paths
- Challenge Design: Your VNFs
  - A design based on a service you will deploy

Flavors and Levels



CP = Connection Point ← VNFD

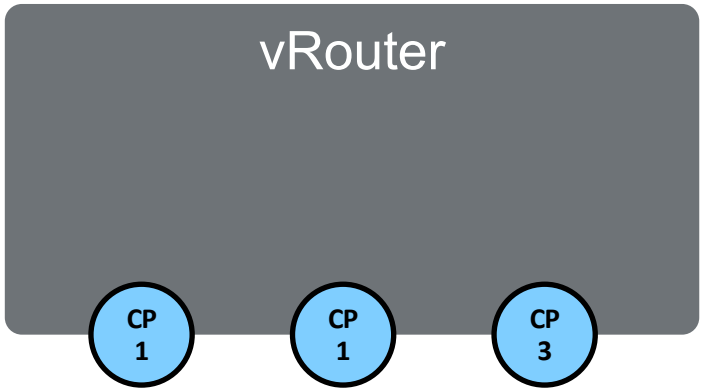
SAP = Service Access Point

VL = Virtual Link ← NSD

VNF Deployment Flavors

- **A flavor contains scaling parameters for achieving horizontal or vertical scaling of a VNF at runtime.**
- **Describes specific requirements for capacity and performance:**
  - Min and Max number of VNFC instances
  - Affinity rules
  - Lifecycle management operations parameters
  - Virtualized resource monitoring parameters
  - Scaling
  - Instantiation levels

VNF Compute Resources



Small	Medium	Large
• RAM: 2048 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 4096 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 8192 • CPU: 4 • SW: vnf_image_id • CPs: cp1, cp2, cp3

Flavor: standard_3_vnfs

Max Bitrate: 500

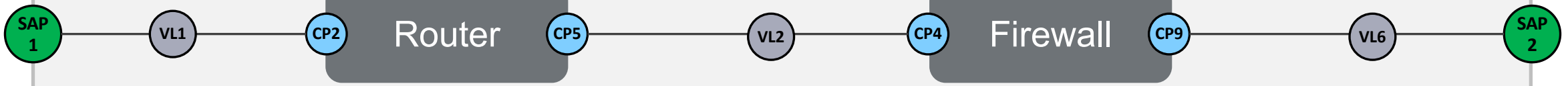
Flavor: small
Instances: 1
Level: 1

Flavor: standard
Instances: 1
Level: 1

Flavor: standard
Instances: 1
Level: 1

Flavor: standard_rtr_fw

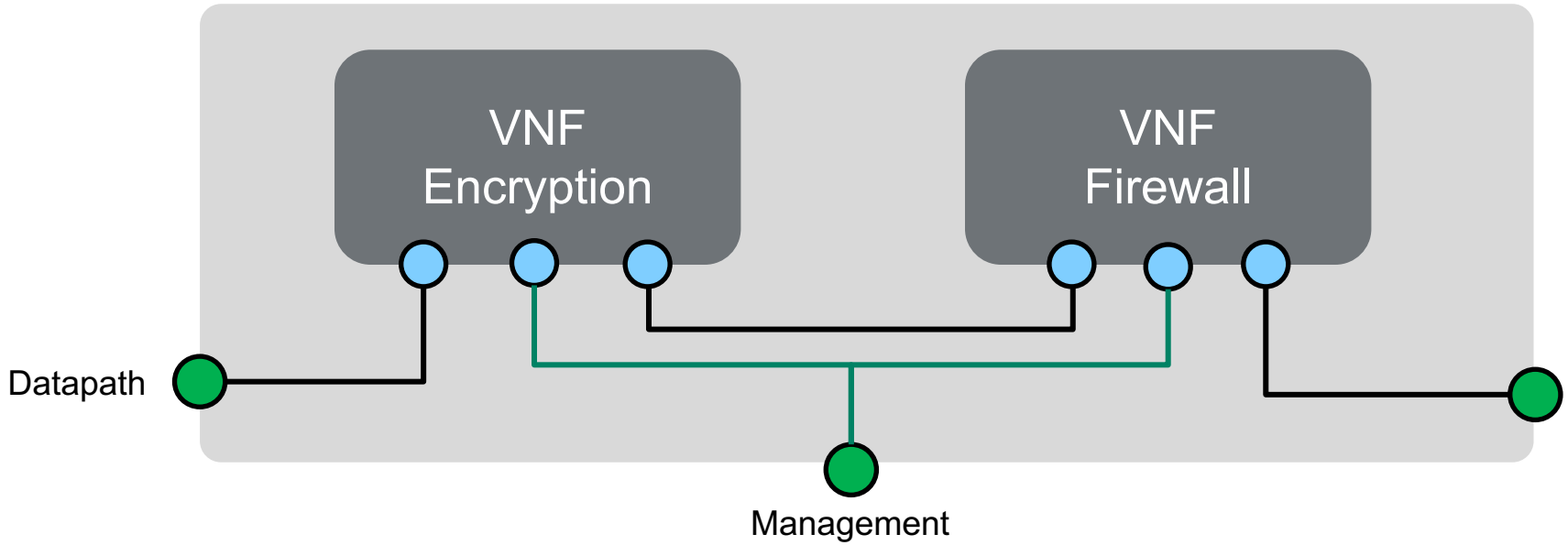
Max Bitrate: 1000



Flavor: standard
Instances: 1
Level: 1

Flavor: standard
Instances: 1
Level: 1

NSD Design #1



What you will build:

- VNFD for Firewall
- NSD with two VNFs
  - 2 flavors: small and medium
  - 1 level

Encryption VNFD will be provided

Onboarding VNF Descriptor Files

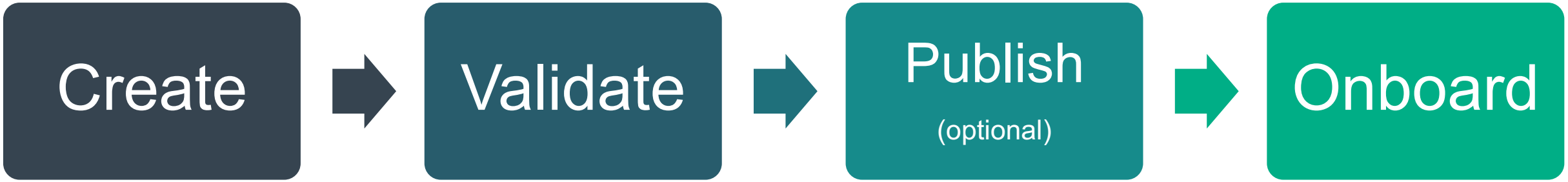
How to Create VNFD Designs

Create a design in Blue Planet

Create JSON and import it to Blue Planet

Upload a CSAR package

Onboarding VNF Designs

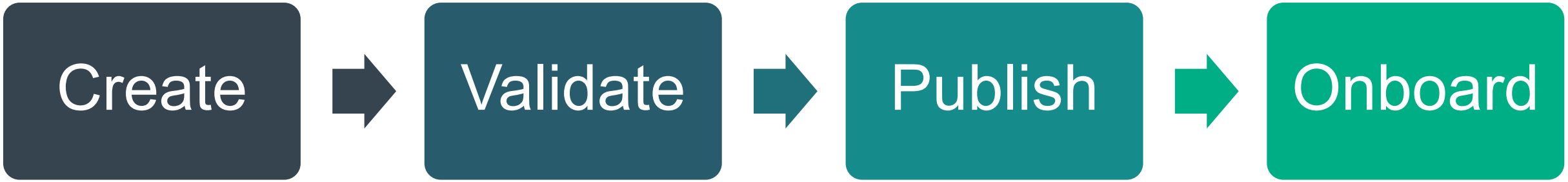


Offboard

Extend

Export

VNFD Creation and Resource Types



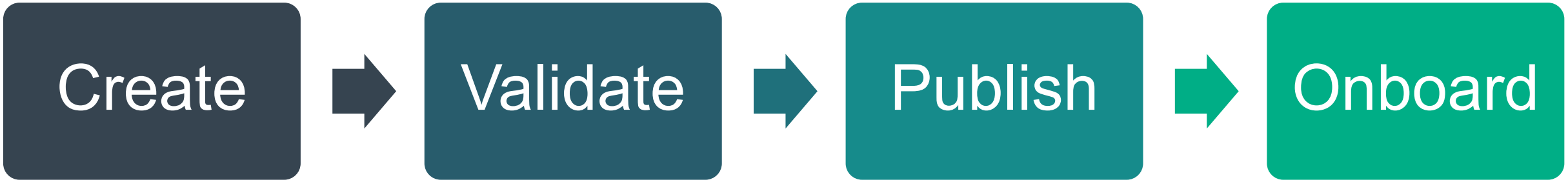
Create function creates a new instance of *nfv2design.resourceTypes.VnfdDesign*

- Design is empty and can be populated by importing JSON or with UI
- Design can be edited or new JSON imported
- Operations are performed on VNFD Design resource – *nfv2design.resourceTypes.VnfdDesign*

Validate verifies the design is complete and consistent.

- JSON is validated during import
- Changes made UI must be validated

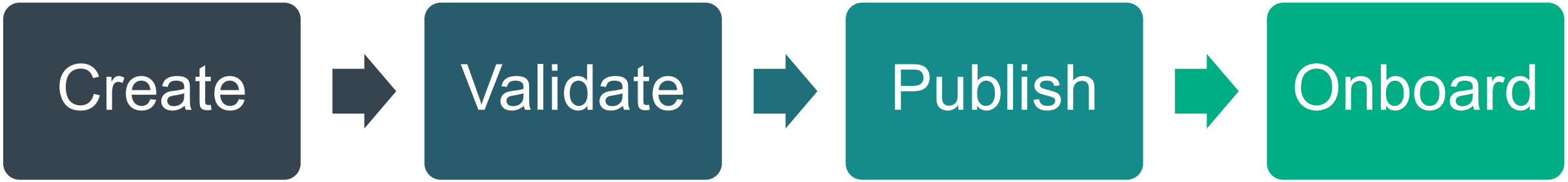
VNFD Creation and Resource Types



Publish marks a design as published and no longer editable

- Design can be exported or extended
- Once published, the design cannot be edited.
  - Offboard operation will not make the design editable

VNFD Creation and Resource Types



Onboard creates the descriptors defined by a design

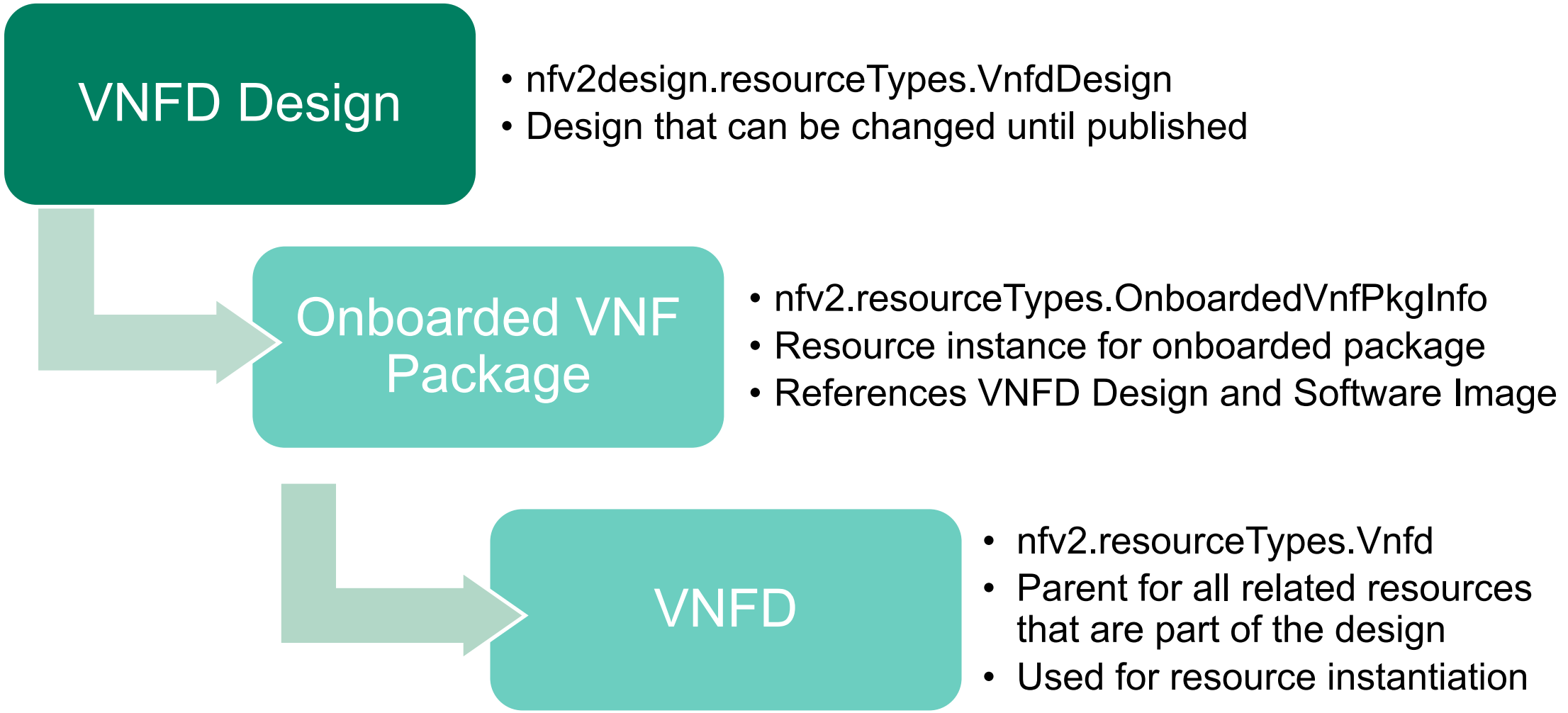
- Image and MD5 file must be on FTP server
- nfv2.resourceTypes.OnboardedVnfPkgInfo
- nfv2.resourceTypes.SwImageDesc
- nfv2.resourceTypes.Vnfd and all sub-resources

Offboard operation on *VnfdDesign* removes descriptors

- *OnboardedVnfPkgInfo* can be deleted, causing all sub-resources to be deleted

Offboard or deleting *OnboardedVnfPkgInfo* does not make the original design editable if it was published

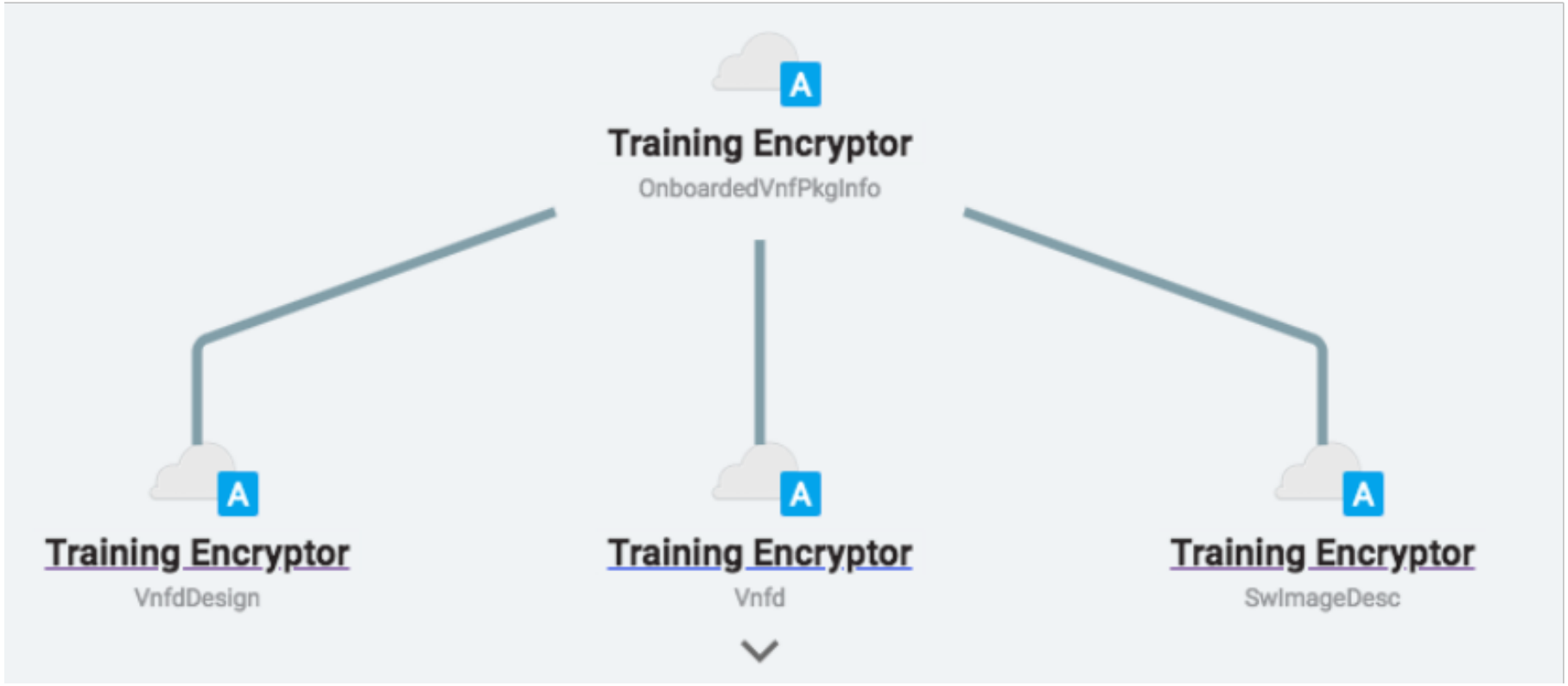
VNF Design Onboarding Summary



What is a VNF Package?

Package = VNF Descriptor + VNF software image

Onboarded VNF Package resource type: nfv2.resourceTypes.OnboardedVnfPkgInfo)



Importing and Exporting VNFD Designs

JSON:

- Import JSON to a design resource
- Export JSON to clipboard to paste into a file
  - Exported JSON is wrapped in a data {} object that needs to be removed.
- Both use **Operations** menu

VNFD Package

- Use **NFVO > Upload/Download VNF package info**
- Download: exports a .zip file that can be uploaded to another BP server
  - Can be used to move VNFD from development environment to production

Uploading VNFD Packages

CSAR: Cloud Service Archive

- Archive with containing VNF definition
- Archive has .zip or .csar extension

BP can import two types of packages:

- BP-NFVO CSAR packages exported from a Blue Planet server
  - Typically this will be exported from a development environment
- ONAP CSAR packages
  - BP-NFVO accepts data model created as per ETSI SOL001
  - ONAP data model is translated into the BP data model during import
  - Uploading an ONAP CSAR results in the creation of a VnfdDesign resource
  - Image and MD5 checksum file should be included in CSAR archive

After uploading a package:

- Add the VNFD to a network service descriptor

Lab 4: Upload a VNF CSAR Package

This lab will be done as an instructor demonstration

Create VNFD Descriptors

Components of VNF Descriptor

Top-level VNFD V1 components:

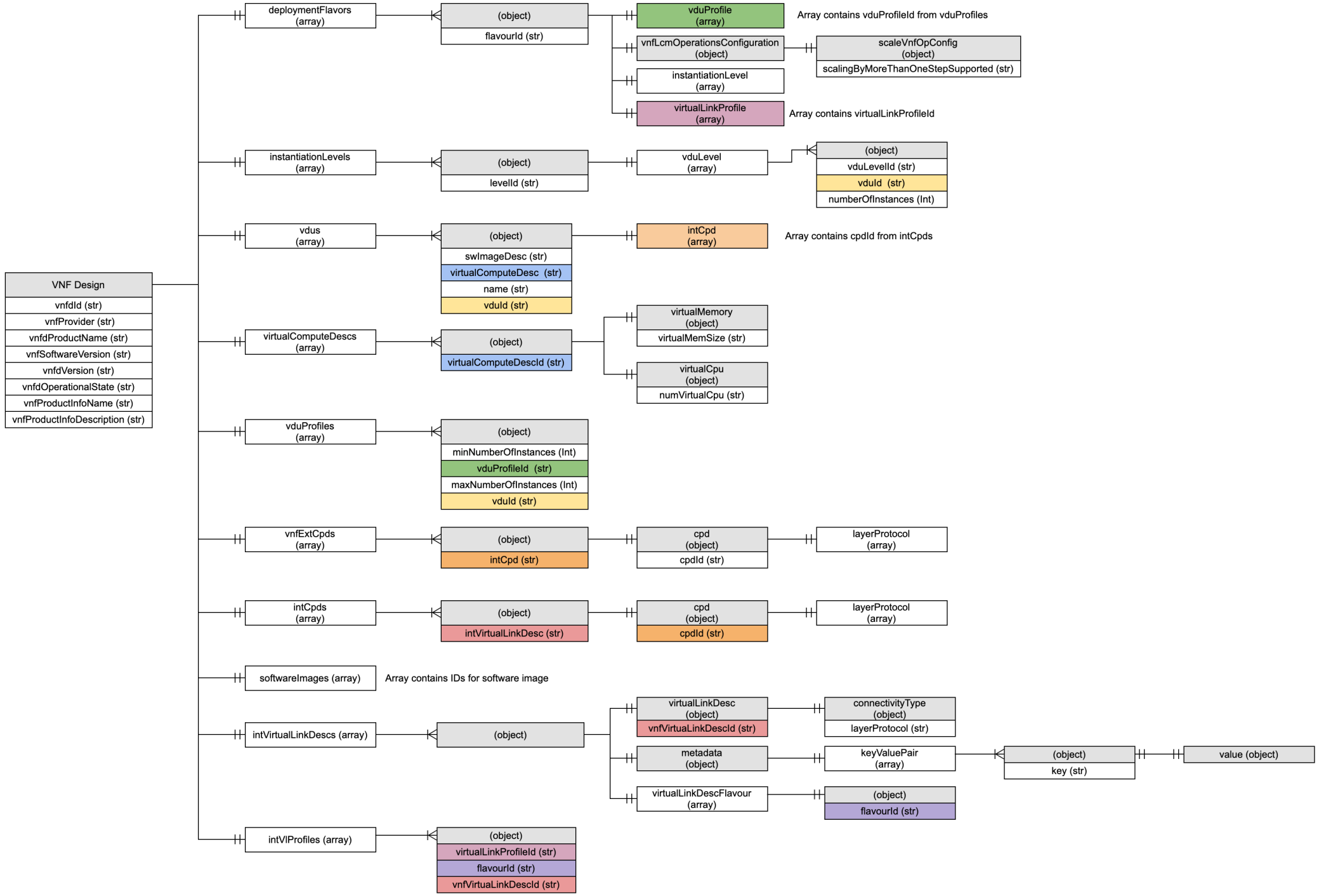
1. VNF Details
2. Software Images
3. Virtual Compute Descriptors
4. Internal Virtual Link Descriptors
5. Internal CPDs (connection points)
6. Virtual Deployment Units
7. External CPDs
8. Instantiation Levels
9. Virtual Deployment Unit Profiles
10. Internal Virtual Link Profiles
11. Deployment Flavors

Top-level VNFD V2 components:

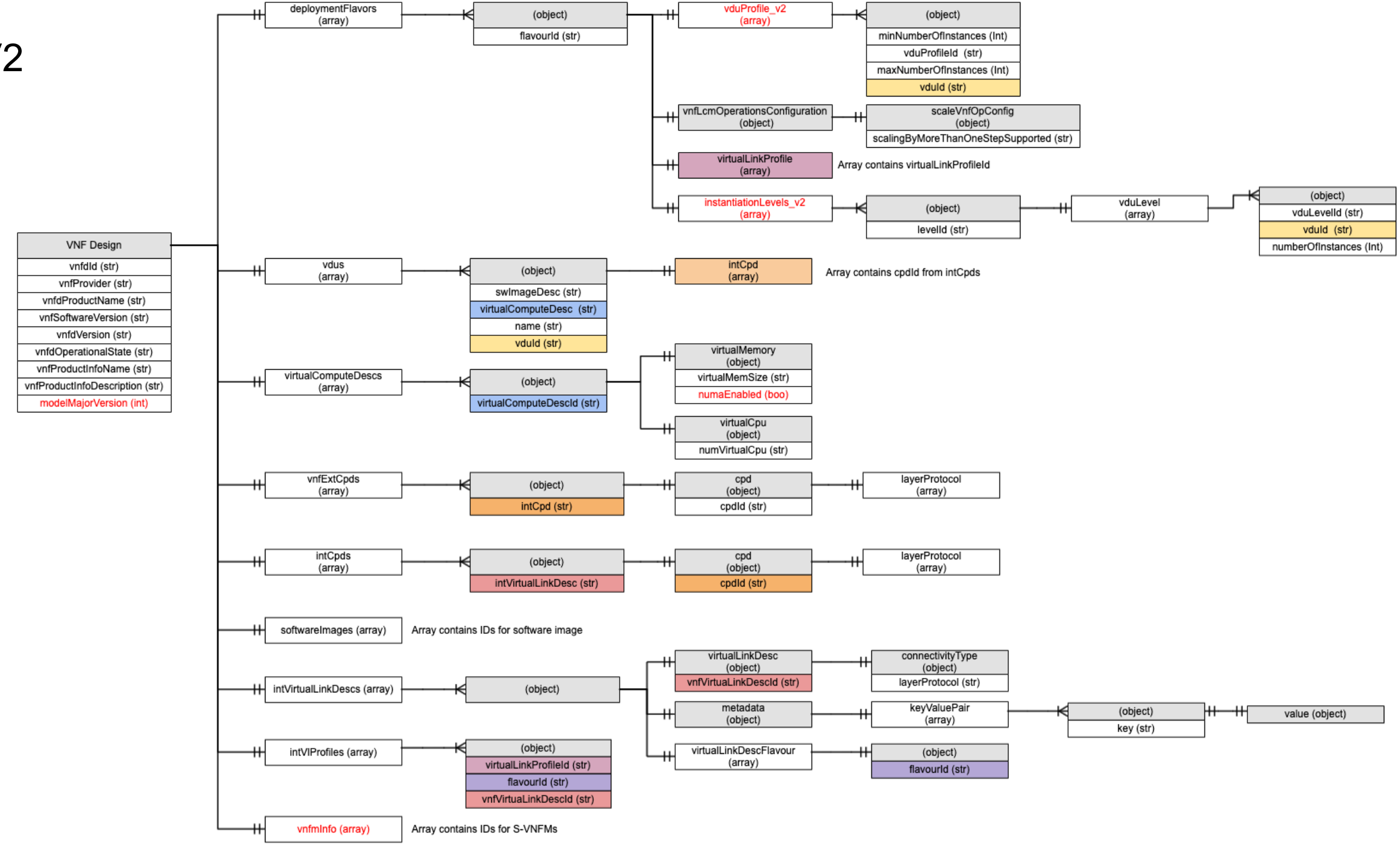
1. VNF Details
2. Software Images
3. Virtual Compute Descriptors
4. Internal Virtual Link Descriptors
5. Internal CPDs (connection points)
6. Virtual Deployment Units
7. External CPDs
8. Internal Virtual Link Profiles
9. Deployment Flavors
10. VIM Info

JSON files do not need to be in this order

V1


VNF Design
vnfdId (str)
vnfProvider (str)
vnfdProductName (str)
vnfSoftwareVersion (str)
vnfdVersion (str)
vnfdOperationalState (str)
vnfProductInfoName (str)
vnfProductInfoDescription (str)
deploymentFlavors (array)
(object)
flavourId (str)
vduProfile (array)
Array contains vduProfileId from vduProfiles
vnfLcmOperationsConfiguration (object)
scaleVnfOpConfig (object)
scalingByMoreThanOneStepSupported (str)
instantiationLevel (array)
virtualLinkProfile (array)
Array contains virtualLinkProfileId
instantiationLevels (array)
(object)
levelId (str)
vduLevel (array)
(object)
vduLevelId (str)
vduId (str)
numberOfInstances (Int)
vdus (array)
(object)
swImageDesc (str)
virtualComputeDesc (str)
name (str)
vduId (str)
intCpd (array)
Array contains cpdId from intCpds
virtualComputeDescs (array)
(object)
virtualComputeDescId (str)
virtualMemory (object)
virtualMemSize (str)
virtualCpu (object)
numVirtualCpu (str)
vduProfiles (array)
(object)
minNumberOfInstances (Int)
vduProfileId (str)
maxNumberOfInstances (Int)
vduId (str)
vnfExtCpds (array)
(object)
intCpd (str)
cpd (object)
cpdId (str)
layerProtocol (array)
intCpds (array)
(object)
intVirtualLinkDesc (str)
cpd (object)
cpdId (str)
layerProtocol (array)
softwareImages (array)
Array contains IDs for software image
intVirtualLinkDescs (array)
(object)
virtualLinkDesc (object)
vnfVirtuaLinkDescId (str)
connectivityType (object)
layerProtocol (str)
metadata (object)
keyValuePair (array)
(object)
key (str)
value (object)
virtualLinkDescFlavour (array)
(object)
flavourId (str)
intVlProfiles (array)
(object)
virtualLinkProfileId (str)
flavourId (str)
vnfVirtuaLinkDescId (str)

V2


VNF Design
vnfdId (str)
vnfProvider (str)
vnfdProductName (str)
vnfSoftwareVersion (str)
vnfdVersion (str)
vnfdOperationalState (str)
vnfProductInfoName (str)
vnfProductInfoDescription (str)
modelMajorVersion (int)
deploymentFlavors (array)
(object)
flavourId (str)
vduProfile_v2 (array)
(object)
minNumberOfInstances (Int)
vduProfileId (str)
maxNumberOfInstances (Int)
vduId (str)
vnfLcmOperationsConfiguration (object)
scaleVnfOpConfig (object)
scalingByMoreThanOneStepSupported (str)
virtualLinkProfile (array)
Array contains virtualLinkProfileId
instantiationLevels_v2 (array)
(object)
levelId (str)
vduLevel (array)
(object)
vduLevelId (str)
vduId (str)
numberOfInstances (Int)
vdus (array)
(object)
swImageDesc (str)
virtualComputeDesc (str)
name (str)
vduId (str)
intCpd (array)
Array contains cpdId from intCpds
virtualComputeDescs (array)
(object)
virtualComputeDescId (str)
virtualMemory (object)
virtualMemSize (str)
numaEnabled (boo)
virtualCpu (object)
numVirtualCpu (str)
vnfExtCpds (array)
(object)
intCpd (str)
cpd (object)
cpdId (str)
layerProtocol (array)
intCpds (array)
(object)
intVirtualLinkDesc (str)
cpd (object)
cpdId (str)
layerProtocol (array)
softwareImages (array)
Array contains IDs for software image
intVirtualLinkDescs (array)
(object)
virtualLinkDesc (object)
vnfVirtualLinkDescId (str)
connectivityType (object)
layerProtocol (str)
metadata (object)
keyValuePair (array)
(object)
key (str)
value (object)
virtualLinkDescFlavour (array)
(object)
flavourId (str)
intVlProfiles (array)
(object)
virtualLinkProfileId (str)
flavourId (str)
vnfVirtualLinkDescId (str)
vnfmInfo (array)
Array contains IDs for S-VNFMs

VNFD Entities

Entity	Resource Type	Label
VNF Design	nfv2design.resourceTypes.VnfdDesign	VNFD Design
Onboarded VNFD	nfv2.resourceTypes.OnboardedVnfPkgInfo	Onboarded Vnf Pkg Info
VNFD	nfv2.resourceTypes.Vnfd	Vnfd
vdus	nfv2.resourceTypes.Vdu	Vdu
virtualComputeDesc	nfv2.resourceTypes.VirtualComputeDesc	Virtual Compute Desc
vduProfiles	nfv2.resourceTypes.VduProfile	Vdu Profile
instantiationLevels	nfv2.resourceTypes.InstantiationLevel	Instantiation Level
deploymentFlavours	nfv2.resourceTypes.VnfDf	Vnf Df
vnfExtCpds	nfv2.resourceTypes.VnfExtCpd	Vnf Ext Cpd
softwareImageDesc	nfv2.resourceTypes.SwImageDesc	Sw Image Desc
intVirtualLinkDescs	nfv2.resourceTypes.VnfVirtualLinkDesc	Vnf Virtual Link Desc
intVlProfiles	nfv2.resourceTypes.VnfVirtualLinkProfile	Vnf Virtual Link Profile

VNFD Details

Basic information about the VNF

- VNF ID: Must be unique in BPO (vnfdId) **(Required)**
- VNF Provider: String, typically the company providing the VNF (vnfProvider) **(Required)**
- VNF Product Name: (vnfdProductName) **(Required)**
- VNF Software Version: String (vnfSoftwareVersion) **(Required)**
- VNFD Version: String **(Required)**
- VNF Product Info Name: String (vnfProductInfoName)
- VNF Description: String (vnfProductInfoDescription)
- Major Model Version: Integer
  - V2 only – not required in V1

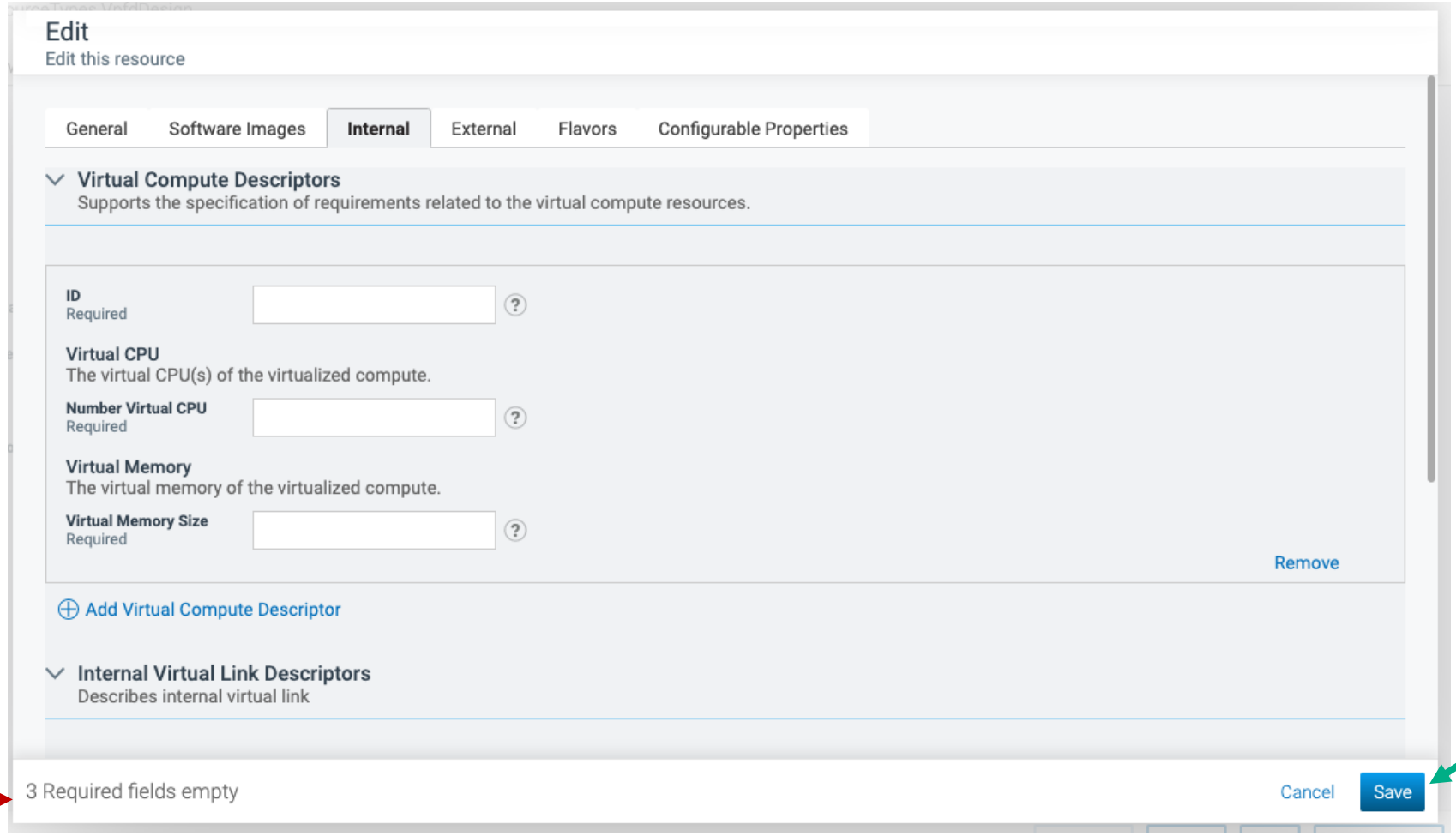
Added to OnboardedVnfPkgInfo resource

Create VNF Design

Go through tabs from left to right

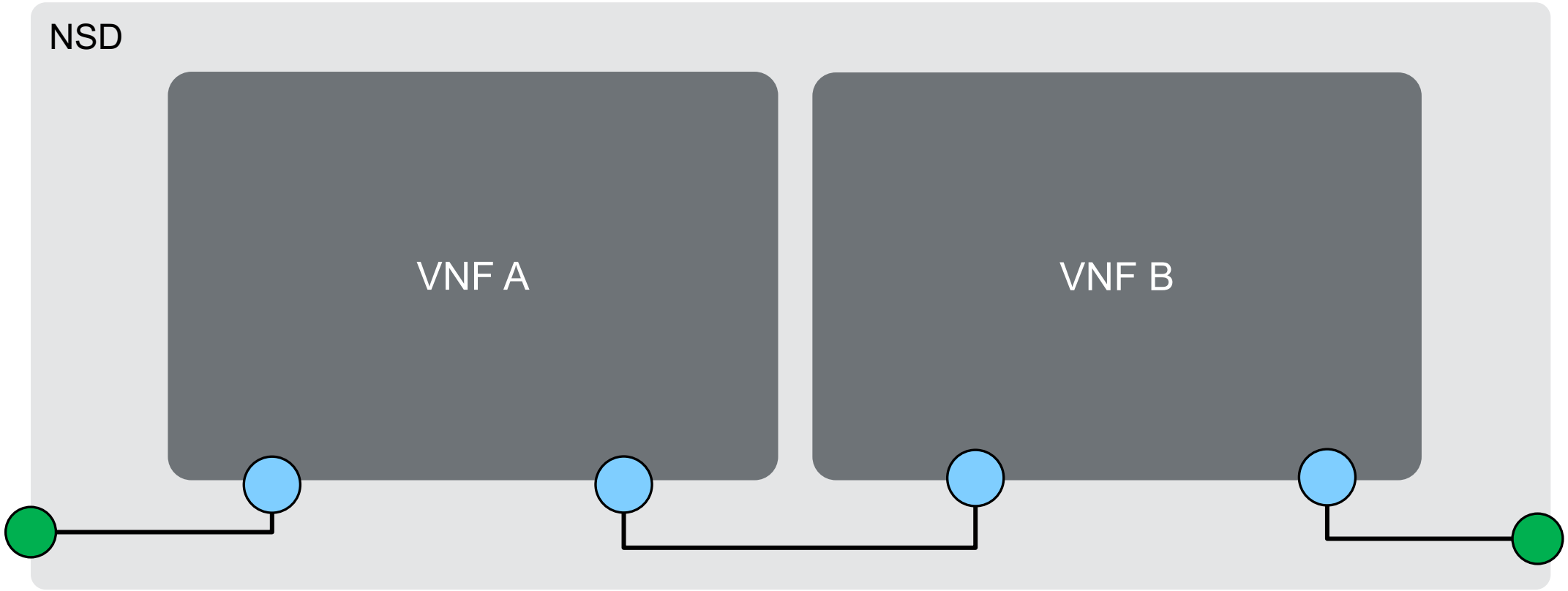
Within a tab, go from top to bottom

Make sure all required fields have entries



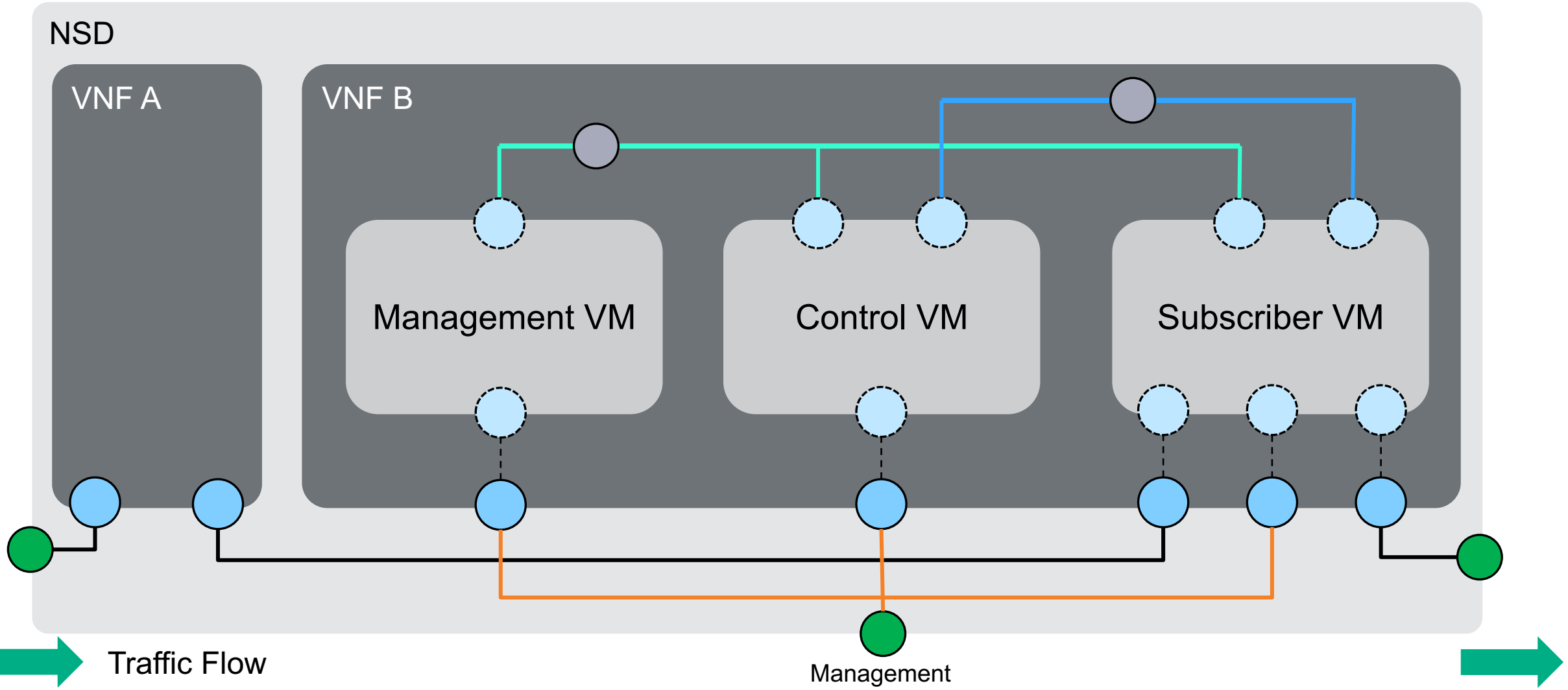
You can save and close anytime

Complex VNF

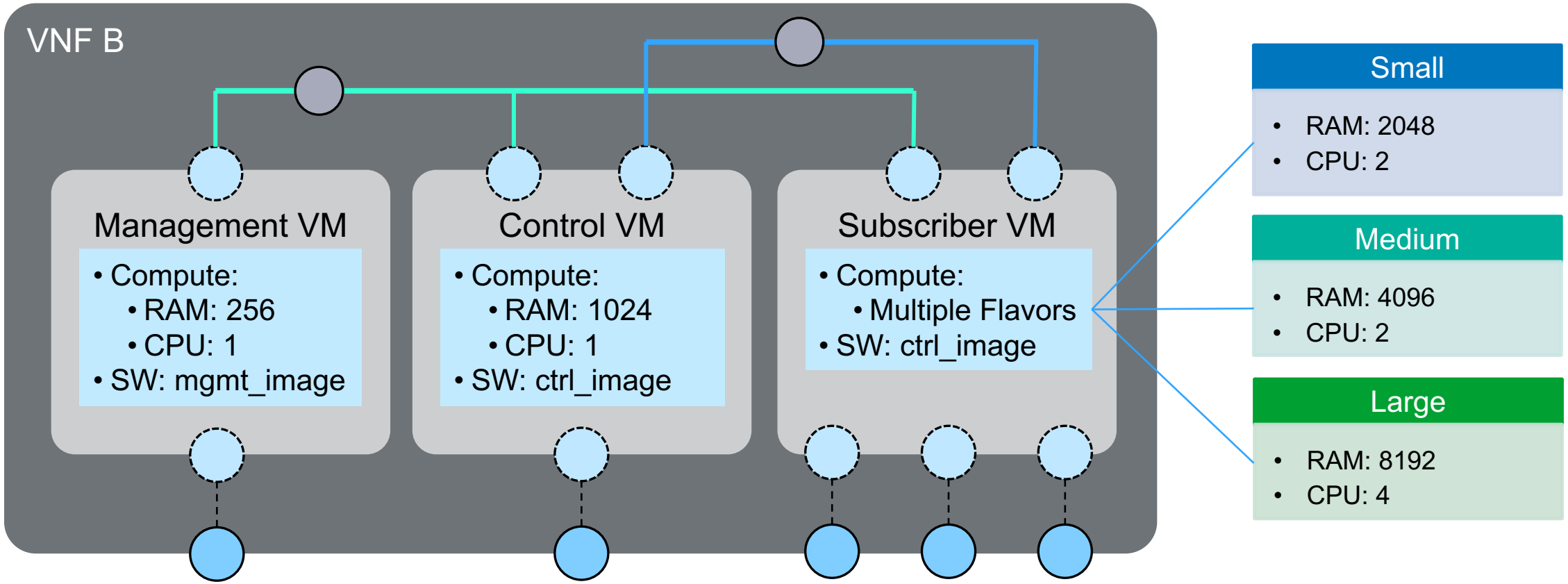


Traffic Flow

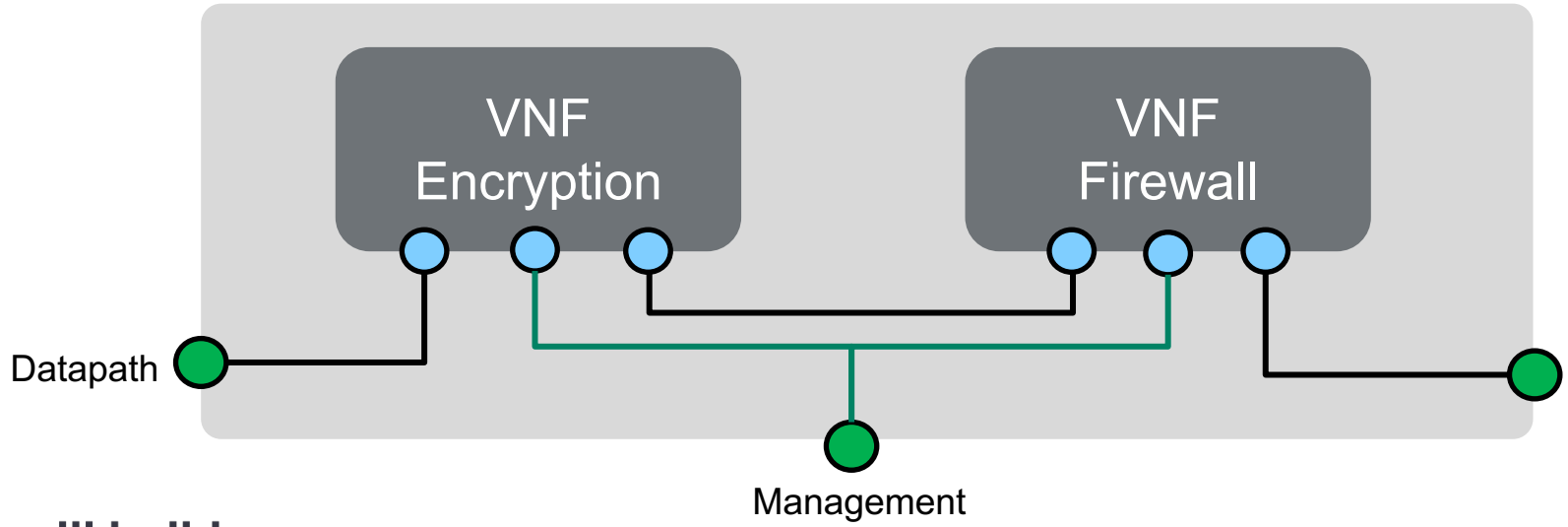
Complex VNF



Defining VM Resources



NSD #1



What you will build:

- VNFD for Firewall
  - Two flavors: small and medium
  - One level
- Import VNFD for encryption
- NSD for service
  - Two flavors
  - One level
  - 3 service access points

Small

- RAM: 1024
- CPU: 2

Medium

- RAM: 2048
- CPU: 2

blueplanet
a division of Ciena

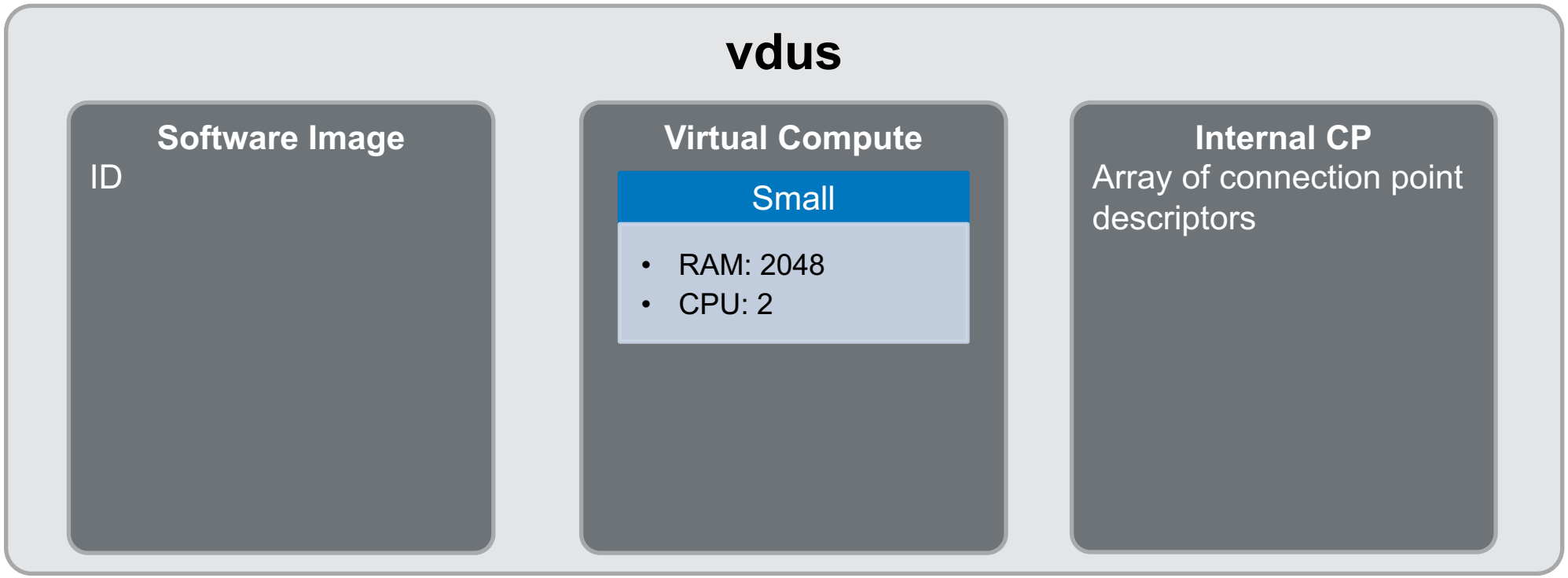
Virtual Deployment Unit (vdu)

Describes the deployment and operational behavior of a VNFC

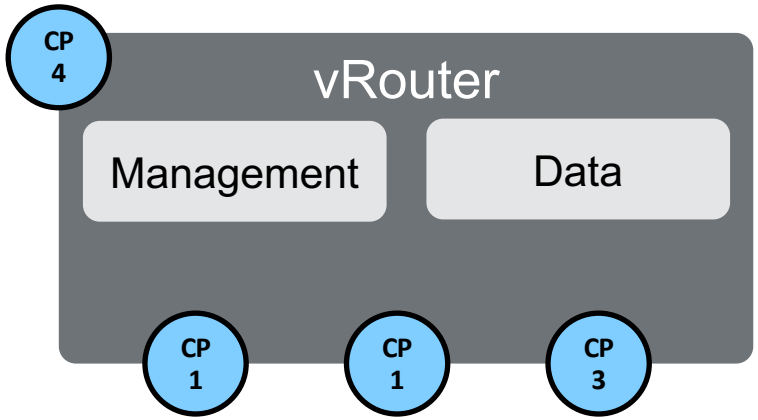
Describes the capabilities of the virtualized containers

- RAM, CPU, Storage, Connectivity

***nfv2.resourceTypes.Vdu* created during onboarding**



Why VDUs Are Needed



Management	Small	Medium	Large
• RAM: 2048 • CPU: 2 • SW: mgmt_img_id • CPs: cp4	• RAM: 2048 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 4096 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 8192 • CPU: 4 • SW: vnf_image_id • CPs: cp1, cp2, cp3

VDU Descriptors

vdus is an array of descriptors

```
"vdus": [
        {
            "intCpd": [
                "fw_mgmt_icpd",
                "fw_left_icpd",
                "fw_right_icpd"
            ],
            "swImageDesc": "firewall_image",
            "virtualComputeDesc": "fw_compute",
            "name": "FW VDU",
            "vduId": "fw_vdu"
        }]
```

softwareImages

Array of software image IDs

IDs will be used to create *nfv2.resourceTypes.SwImageDesc* resources during onboarding

- ID will be used as the BP Label for the resource
- ID will one of the resource properties.
- BP resource will have a BP GUID like any other resource

Virtual Compute Descriptors (virtualComputeDescs)

Array of descriptors
ID is used in VDU descriptor
***nfv2.resourceTypes.VirtualComputeDesc* resource for each descriptor is created during onboarding**

```
"virtualComputeDescs": [
        {
            "virtualMemory": {
                "virtualMemSize": 256,
                "numaEnabled": false
            },
            "virtualComputeDescId": "fw_compute",
            "virtualCpu": {
                "numVirtualCpu": 1
            }
        }]
```

Used in VDU descriptor

Internal Virtual Link Descriptors (intVirtualLinkDescs)

Array of descriptors

ID is used to connect intCpd to the link

```
"intVirtualLinkDescs": [{
    "virtualLinkDesc": {
        "virtualLinkDescId": "base_vld",
         "connectivityType": {
              "layerProtocol": "IPV4"
          }
      },
      "virtualLinkDescFlavour": [{
          "flavourId":"base_vld_flavour"
        }],
      "metadata": { }
    }
```

Used in intCpd and intVlProfiles descriptors

Used in intVlProfiles descriptors

Virtual Link Descriptors JSON Example

```
intVirtualLinkDescs: {
      "virtualLinkDescId": "base_vld",
      "connectivityType": {
          "layerProtocol": "IPV4"
          }
       },
      "virtualLinkDescFlavour": [{
            "flavourId":"base_vld_flavour"
       }]
      }
```

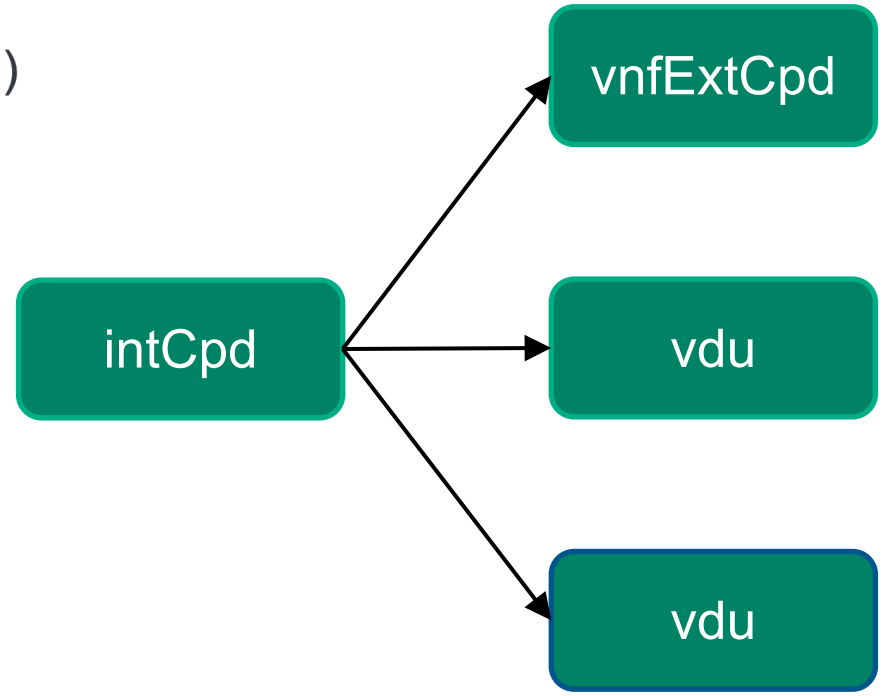
Internal Connection Points (intCpds)

Connectivity is modeled through:

- VDU Connection Point Descriptors (VduCpd)
- VNF External Connection Point Descriptors (VnfExternalCpd)
- Internal Virtual Link Descriptors (intVirtualLinkDescs)

Start by defining intCpds:

```
"intCpds": [{
   "cpd": {
      "layerProtocol": [ "IPv4" ],
      "cpdId": "csr_lan_gi3_icpd"
    },
   "intVirtualLinkDesc": "base_vld",
}]
```



- intVirtualLinkDesc is optional
- *nfv2.resourceTypes.VduCpd* resource for each **cpd** is created during onboarding

Connection Point Descriptor (cpd) Properties

layer_protocol:

- String
- Required
- Valid values: Ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire

role:

- String
- Valid values: root, leaf

description

- String

address_data

- Type: AddressData[ ]
  - Properties:
    - address_type: Required; valid values: *mac_address* or *ip_address*
    - Optional: l2_address_data, l3_address_data data types
- Provides information on the addresses to be assigned to the connection point(s) instantiated from this Connection Point Descriptor.

External Connection Point Descriptors (vnfExtCpds)

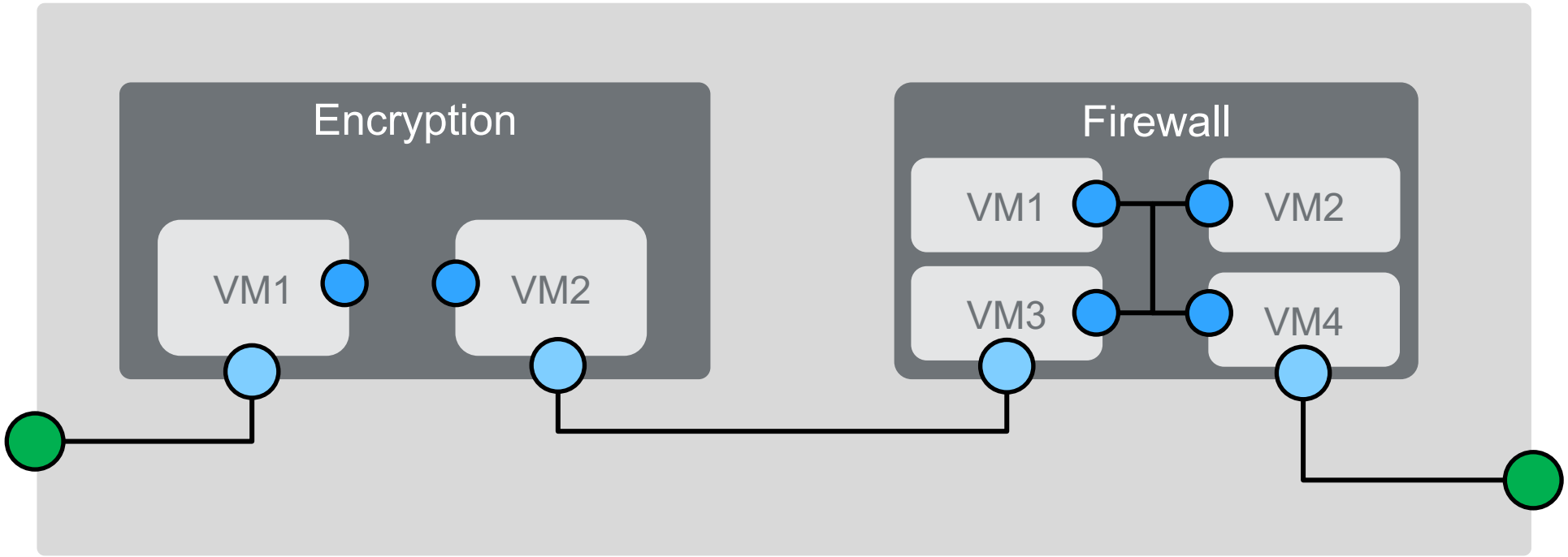
Array of descriptors

vnfExtCpds references the ID of an intCpd descriptor

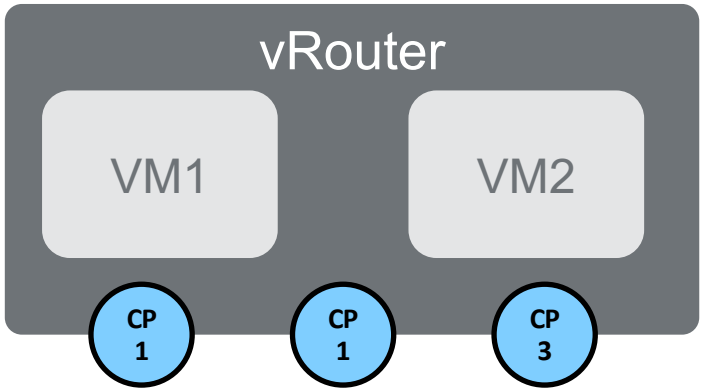
```
"vnfExtCpds": [{
            "intCpd": "fw_mgmt_icpd",
            "cpd": {
                "layerProtocol": [
                    "IPv4"
                ],
                "cpdId": "fw_mgmt_extcpd"
            }
        }]
```

***nfv2.resourceTypes.VnfExtCpd* resource for each member is created during onboarding**

Why do we need internal and external CPs?



VNF Instantiation Levels



Level 1: 1 compute instance
Level 2: 2 compute instances

Small	Medium	Large
• RAM: 2048 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 4096 • CPU: 2 • SW: vnf_image_id • CPs: cp1, cp2, cp3	• RAM: 8192 • CPU: 4 • SW: vnf_image_id • CPs: cp1, cp2, cp3

Instantiation level defines the VDU and number of instances

Instantiation Levels (instantiationLevels)

Array of descriptors

***nfv2.resourceTypes.InstantiationLevel* resource for each descriptor is created during onboarding**

For complex VNFs multiple vduLevels may be needed

```
"instantiationLevels": [
        {
            "levelId": "fw_level",
            "vduLevel": [{
                "vduLevelId": "fw_vdu_level",
                "vduId": "fw_vdu",
                "numberOfInstances": 1
            }]
        }]
```

vduId is the same as the vduId in the corresponding VDU descriptor in vdus.

Deployment Flavors

Array of descriptors

***nfv2.resourceTypes.VnfDf* resource for each descriptor is created during onboarding**

Contains:

- Virtual Link Profiles – list of internal VL IDs used in the flavor (optional, for complex VNFs)
- VDU Profiles – list of VDU IDs profiles for each VNFC used in the flavor
  - Complex VNFs will have multiple VDU profiles IDs
- Flavor ID – Unique identifier of the flavor
- Instantiation Levels – List of IDs for all levels used in the flavor
- Default instantiation level – must be set if you have multiple levels

VDU Profiles (vduProfiles)

Array of descriptors

***nfv2.resourceTypes.VduProfile* resource for each descriptor is created during onboarding**

Used in a flavor to limit the number of VNFC instances that can be created

```
"vduProfiles": [
        {
            "minNumberOfInstances": 1,
            "vduProfileId": "fw_vdu_profile",
            "maxNumberOfInstances": 1,
            "vduId": "fw_vdu"
        }]
```

vduId is the same as the vduId in the corresponding VDU descriptor in vdus.

Deployment Flavors

```
"deploymentFlavors": [{
      "virtualLinkProfile": [
             "base_vld_profile",
              "ew_vld_profile"
          ],
      "vduProfile": [
          "fw_vdu_profile"
       ],
       "vnfLcmOperationsConfiguration": {
          "scaleVnfOpConfig": {
             "scalingByMoreThanOneStepSupported": false
          }
       },
       "flavourId": "fw_flavor",
       "instantiationLevel": [
          "fw_level1", "fw_level2"
       ]
      "defaultInstantiationLevel": "fw_level1"
   }]
```

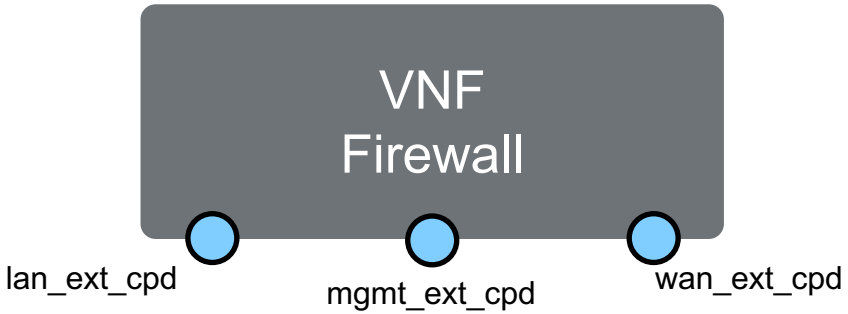
Reference to VL profiles
(only used with complex VNF)

Reference to VDU profile

Reference to instantiation level

Lab 5: Create a VNFD

- **Create VNF Design for the Firewall VNF**
- **Validate the VNF design**



Flavors

Small

- RAM: 1024
- CPU: 2

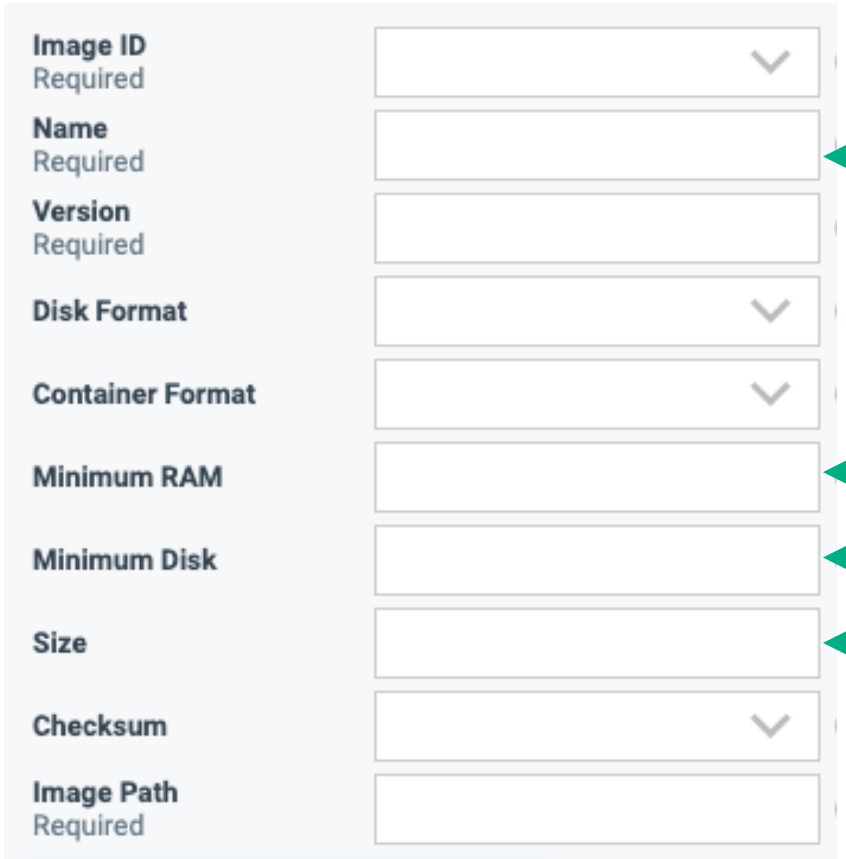
Medium

- RAM: 2048
- CPU: 2

Onboard a VNF Design

Onboard creates the descriptors defined by a design

VNF Image and .md5 checksum file must be upload to SFTP server



Only alphanumeric and _ (underscore) for Ciena DNFVI

Value in megabytes

Value in bytes, should be exact size of file

Value in bytes, can exceed actual size

FTP Server

One FTP server (internal or external) is supported at a time.

NFVO microservice automatically creates an internal FTP server.

- Credentials: admin/adminpw
- Port: 2023

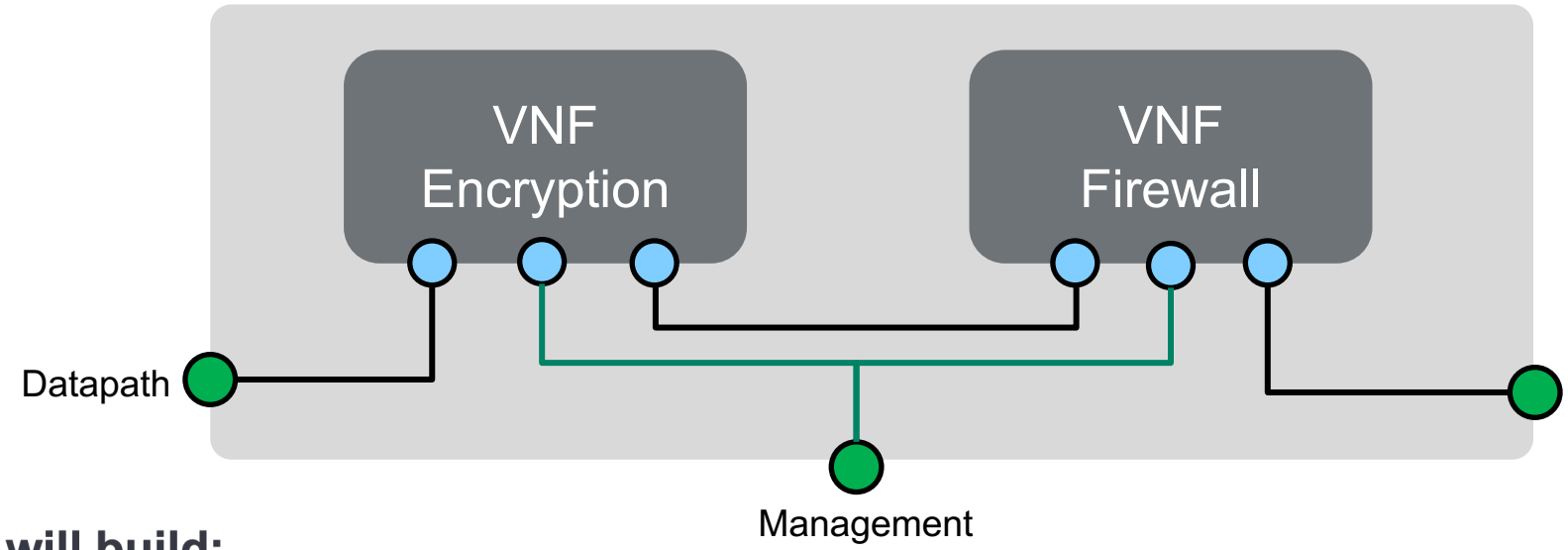
Images are uploaded to the /nfvo directory.

- Uploading may take several minutes; wait for the prompt to return.

To onboard a VNF image, it must have an associated MD5 sum file or the download to the VIM fails.

Lab 6: Onboard a VNFD Package

NSD #1

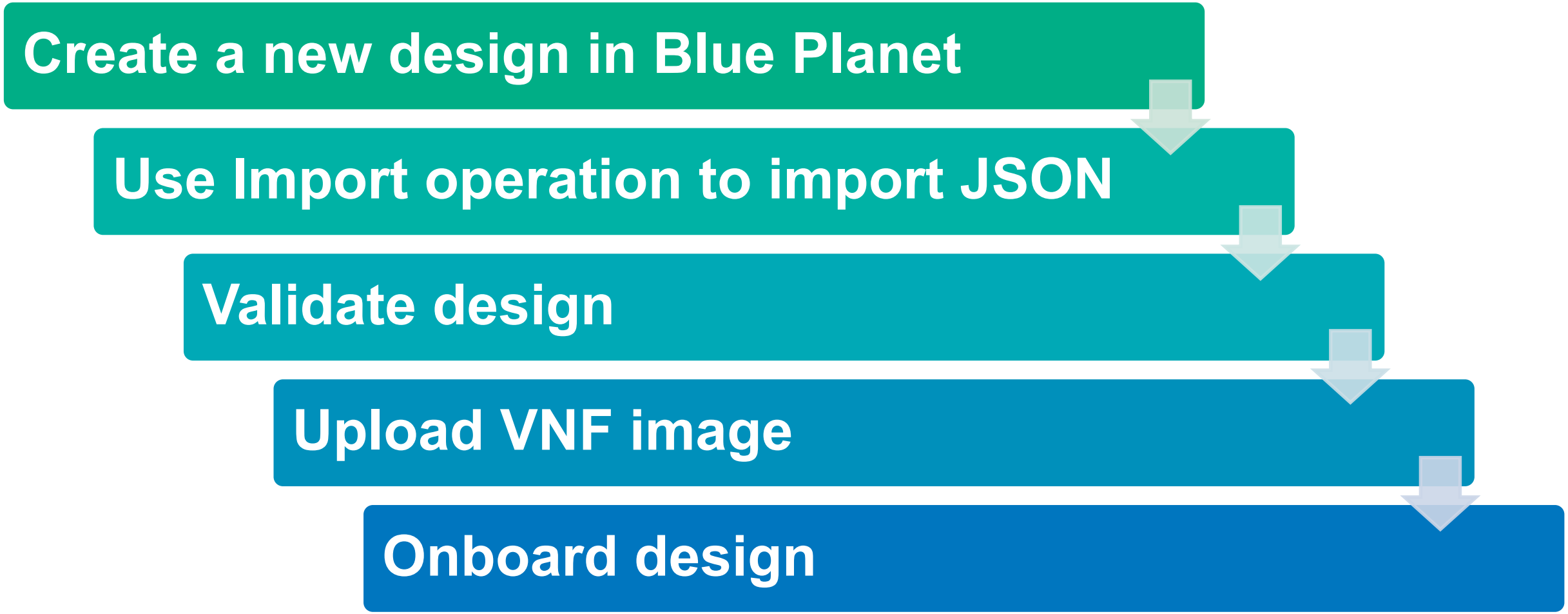


What you will build:

- ✓ VNFD for Firewall
  - Two flavors: small and medium
  - One level
- Import VNFD for encryption ← What we need to do next
- NSD for service
  - Two flavors
  - One level
  - 3 service access points

blueplanet
a division of Ciena

Importing VNFD Designs



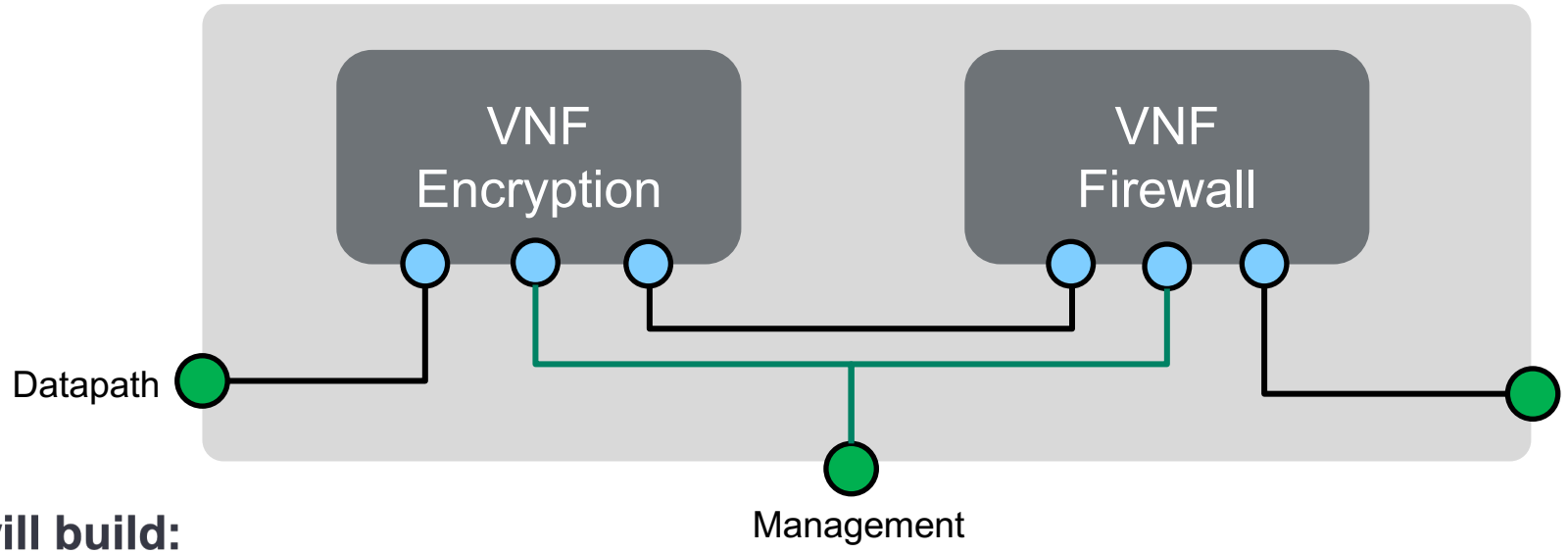
blueplanet
a division of Ciena

Lab 7: Import JSON

Challenge Lab: Create VNFD using JSON

Create a design using JSON
Import the design
Create an MD5 checksum
Upload the VNF image and MD5 checksum
Onboard the design

NSD #1



What you will build:

- ✓ VNFD for Firewall
  - Two flavors: small and medium
  - One level
- ✓ Import VNFD for encryption
- NSD for service ← What we need to do next
  - Two flavors
  - One level
  - 3 service access points

Network Service Descriptors

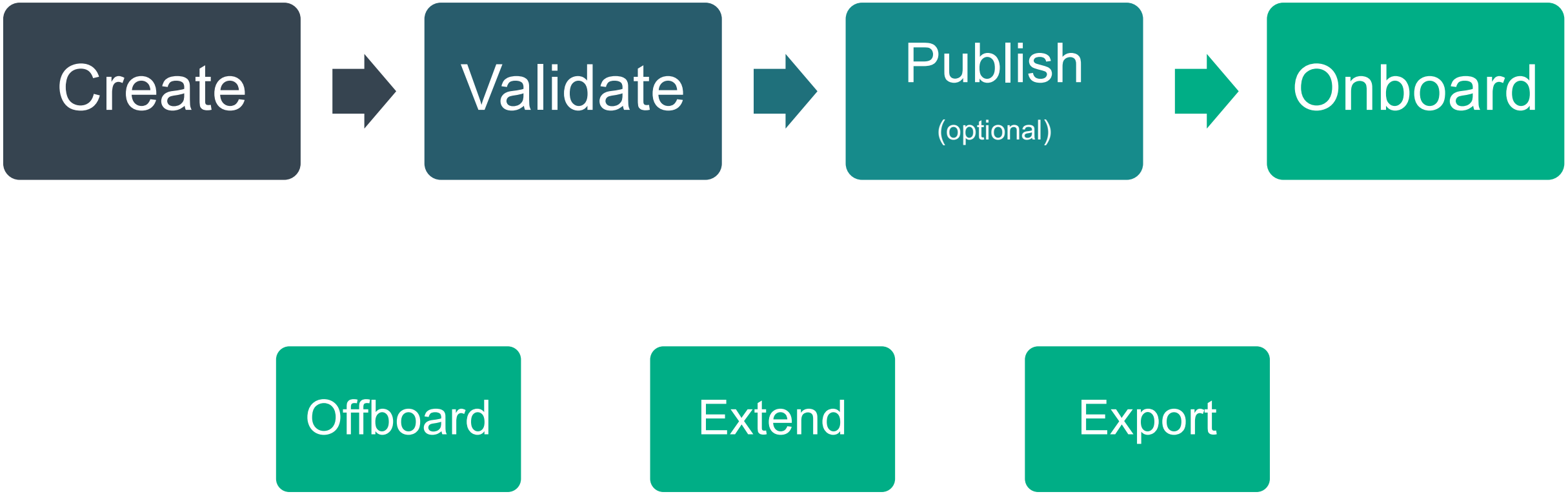
How to Create NSD Designs

Create a design in Blue Planet

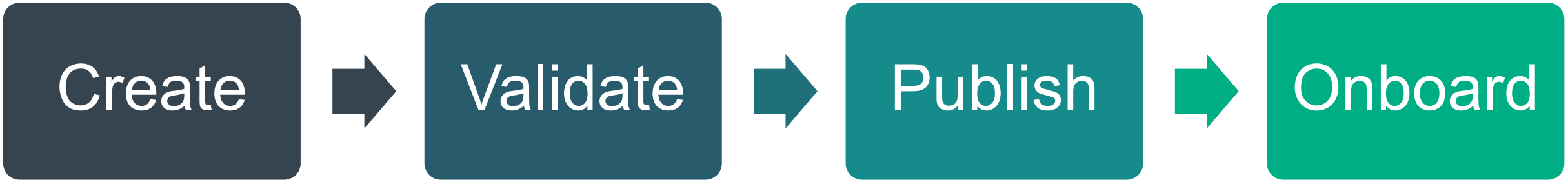
Create JSON and import it to Blue Planet

Upload a NSD package

NSD Design Operations



NSD Creation and Resource Types



Create function creates a new instance of *nfv2design.resourceTypes.NsdDesign*

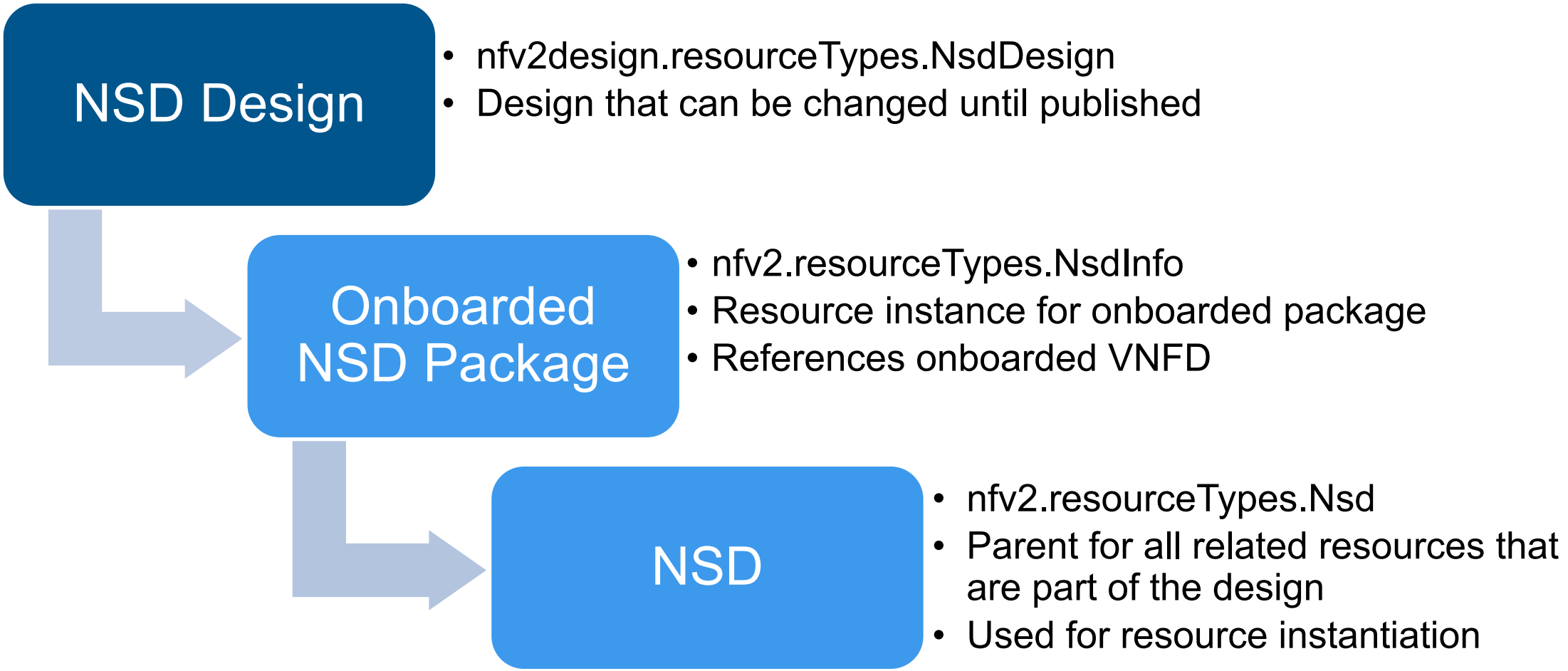
Validate verifies the design is complete and consistent.

Publish marks a design as published and no longer editable

Onboard:

- VNF designs used in NSD must be onboarded before onboarding NSD
- Creates *nfv2.resourceTypes.NsdInfo* resource
- Creates *nfv2.resourceTypes.*Nsd resource and sub-resources
- Creates a relationship between *NsdInfo* resource and *OnboardedVnfPkgInfo* resources for VNFs

NSD Design Onboarding



Network Service Descriptor Requirements

Before onboarding a Network Service Design:

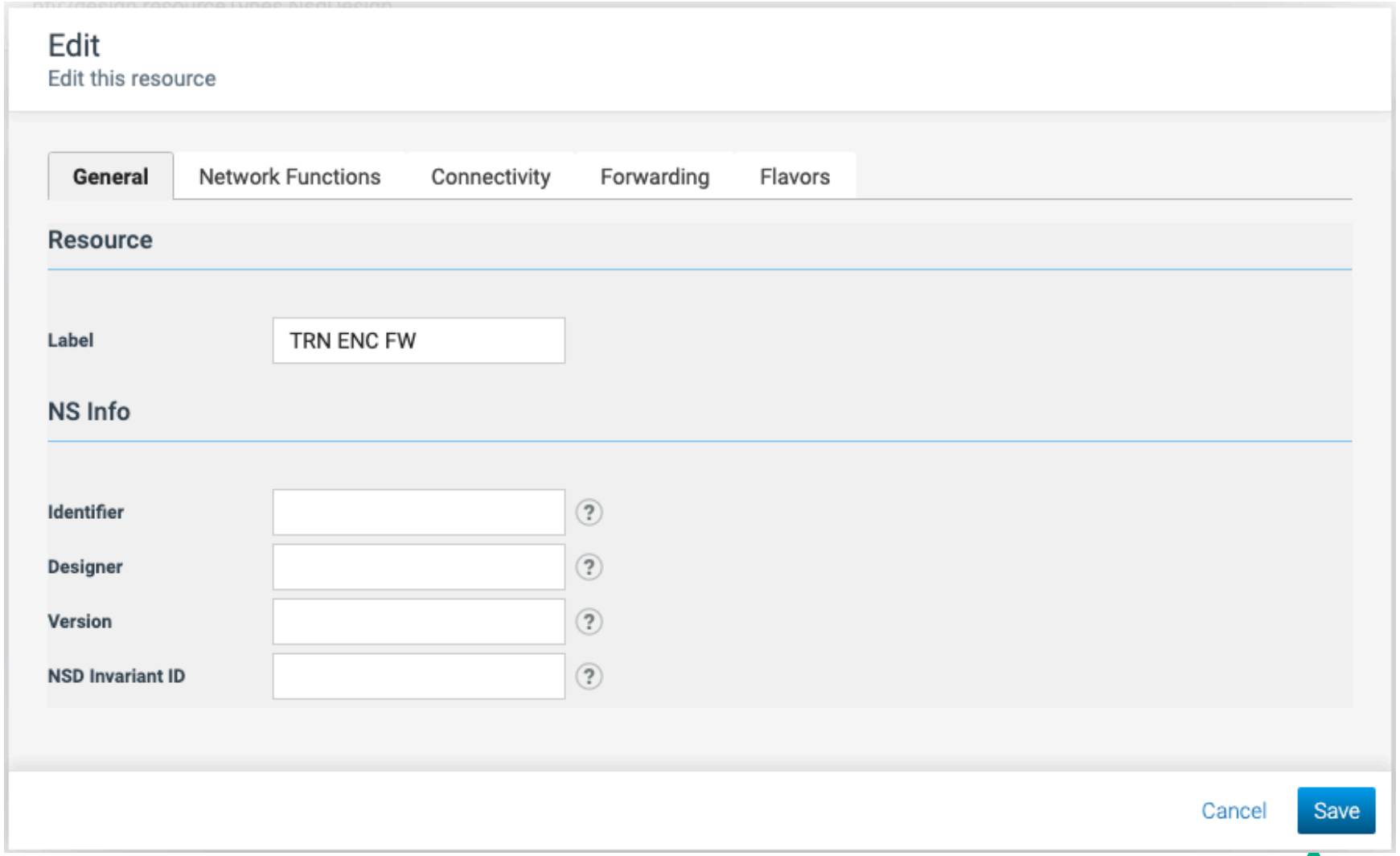
- VNF Designs for all VNFs included in the network service must be onboarded
- VNF images must be uploaded

Create NSD Design

Go through tabs from left to right

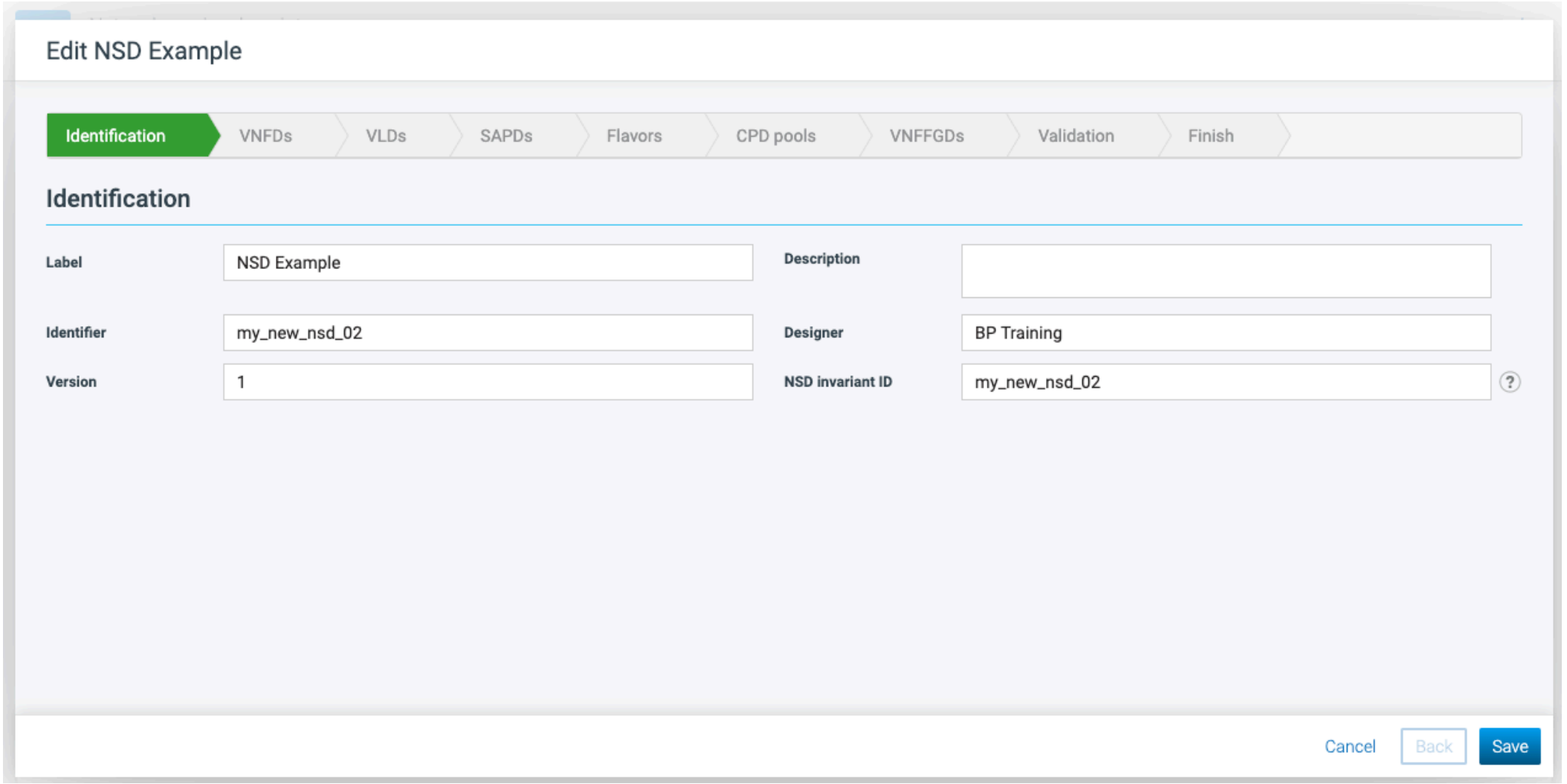
Within a tab, go from top to bottom

Make sure all required fields have entries

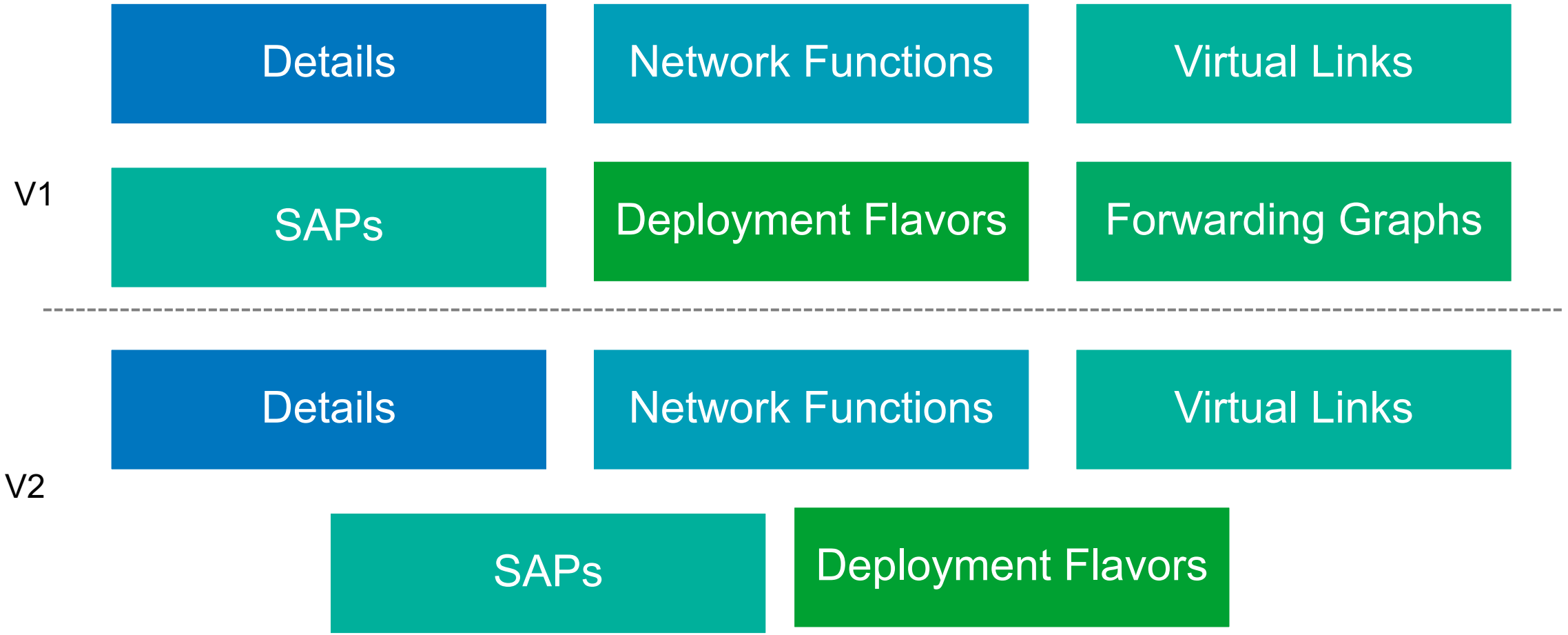


You can save and close anytime

NSD Design Builder



Network Service Descriptors



Components of a Network Service Design

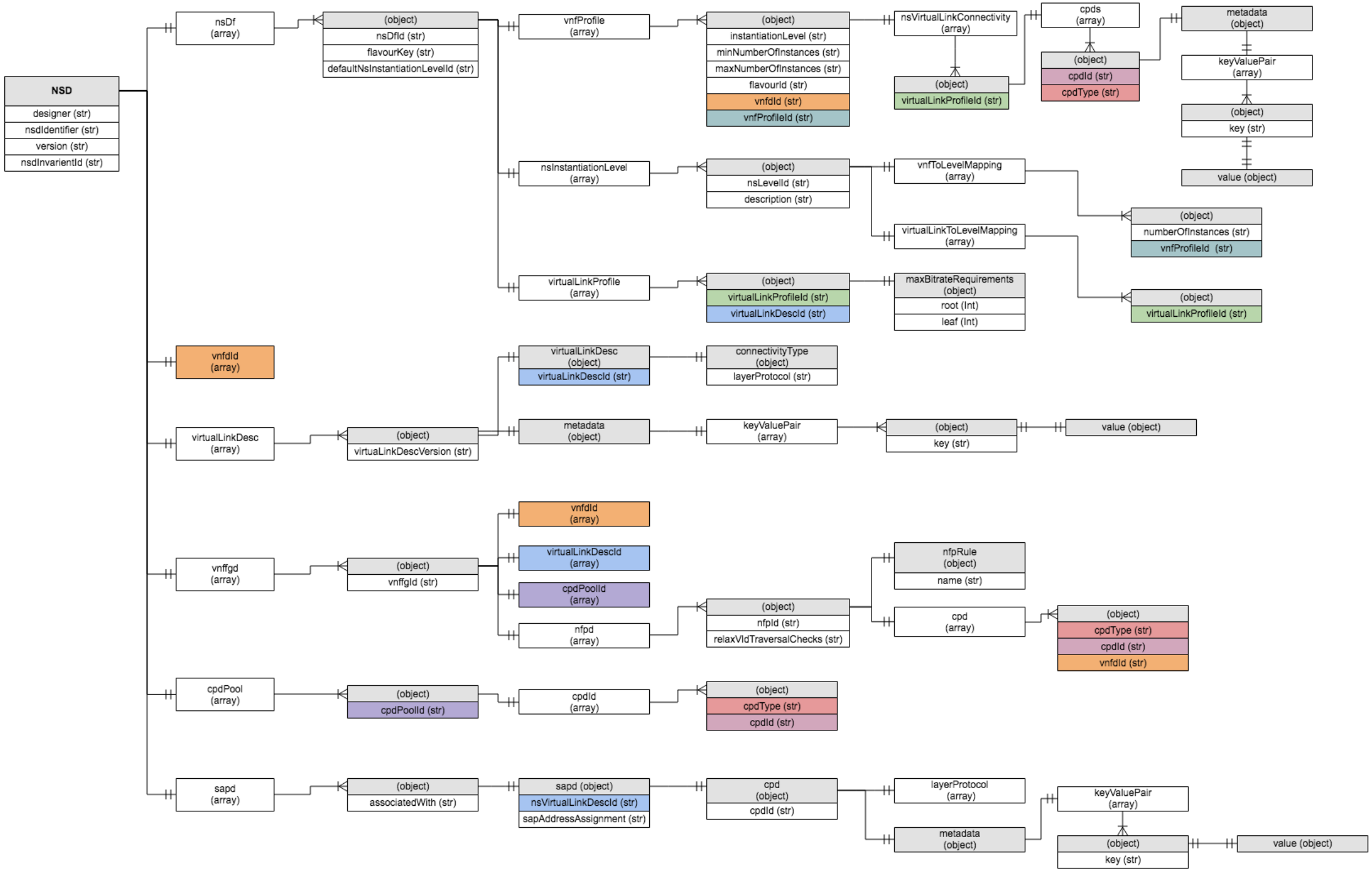
V1

- **Details:**
  - nsdIdentifier – String (required)
  - designer – String (required)
  - version – String (required)
  - nsdInvariantId – String (required)
  - exportedModelVersion - String
- **Network Functions:**
  - vnfdId - Array
- **Connectivity:**
  - virtualLinkDesc – Array
  - sapd – Array
- **Flavors:**
  - nsDf – Array
- **Forwarding:**
  - vnffgd – Array
  - cpdPool – Array

V2

- **Details:**
  - nsdIdentifier – String (required)
  - designer – String (required)
  - version – String (required)
  - nsdInvariantId – String (required)
  - exportedModelVersion – String
  - MajorModelVersion - Integer
- **Network Functions:**
  - vnfdId - Array
- **Connectivity:**
  - virtualLinkDesc – Array
  - sapd – Array
- **Flavors:**
  - nsDf – Array

V1


NSD
designer (str)
nsdIdentifier (str)
version (str)
nsdInvarientId (str)
nsDf (array)
(object)
nsDfId (str)
flavourKey (str)
defaultNsInstantiationLevelId (str)
vnfProfile (array)
(object)
instantiationLevel (str)
minNumberOfInstances (str)
maxNumberOfInstances (str)
flavourId (str)
vnfdId (str)
vnfProfileId (str)
nsVirtualLinkConnectivity (array)
(object)
virtualLinkProfileId (str)
cpds (array)
(object)
cpdId (str)
cpdType (str)
metadata (object)
keyValuePair (array)
(object)
key (str)
value (object)
nsInstantiationLevel (array)
(object)
nsLevelId (str)
description (str)
vnfToLevelMapping (array)
(object)
numberOfInstances (str)
vnfProfileId (str)
virtualLinkToLevelMapping (array)
(object)
virtualLinkProfileId (str)
virtualLinkProfile (array)
(object)
virtualLinkProfileId (str)
virtualLinkDescId (str)
maxBitrateRequirements (object)
root (Int)
leaf (Int)
vnfdId (array)
virtualLinkDesc (array)
(object)
virtualLinkDescVersion (str)
virtualLinkDesc (object)
virtualLinkDescId (str)
connectivityType (object)
layerProtocol (str)
metadata (object)
keyValuePair (array)
(object)
key (str)
value (object)
vnffgd (array)
(object)
vnffgId (str)
vnfdId (array)
virtualLinkDescId (array)
cpdPoolId (array)
nfpd (array)
(object)
nfpId (str)
relaxVldTraversalChecks (str)
nfpRule (object)
name (str)
cpd (array)
(object)
cpdType (str)
cpdId (str)
vnfdId (str)
cpdPool (array)
(object)
cpdPoolId (str)
cpdId (array)
(object)
cpdType (str)
cpdId (str)
sapd (array)
(object)
associatedWith (str)
sapd (object)
nsVirtualLinkDescId (str)
sapAddressAssignment (str)
cpd (object)
cpdId (str)
layerProtocol (array)
metadata (object)
keyValuePair (array)
(object)
key (str)
value (object)

Entity	Resource Type	Label
Nsd Design	nfv2design.resourceTypes.NsdDesign	NSD Design
Nsd Info	nfv2.resourceTypes.NsdInfo	Nsd Info
Nsd	nfv2.resourceTypes.Nsd	Nsd
vnfd	nfv2.resourceTypes.Vnfd	Vnfd
virtualLinkDesc	nfv2.resourceTypes.NsVirtualLinkDesc	Ns Virtual Link Desc
vnffgd	nfv2.resourceTypes.Vnffgd	Vnffgd
cpdPool	nfv2.resourceTypes.CpdPool	Cpd Pool
nsDf	nfv2.resourceTypes.NsDf	NS DF
sapd	nfv2.resourceTypes.Sapd	Sapd
nfpd	nfv2.resourceTypes.Nfpd	Nfpd
vnfProfile	nfv2.resourceTypes.VnfProfile	Vnf Profile
nsInstantiationLevel	nfv2.resourceTypes.NsLevel	Ns Level
virtualLinkProfile	nfv2.resourceTypes.NsVirtualLinkProfile	Ns Virtual Link Profile
vnf ext cpd	nfv2.resourceTypes.VnfExtCpd	Vnf Ext Cpd

NSD Details

Provides general information about the design

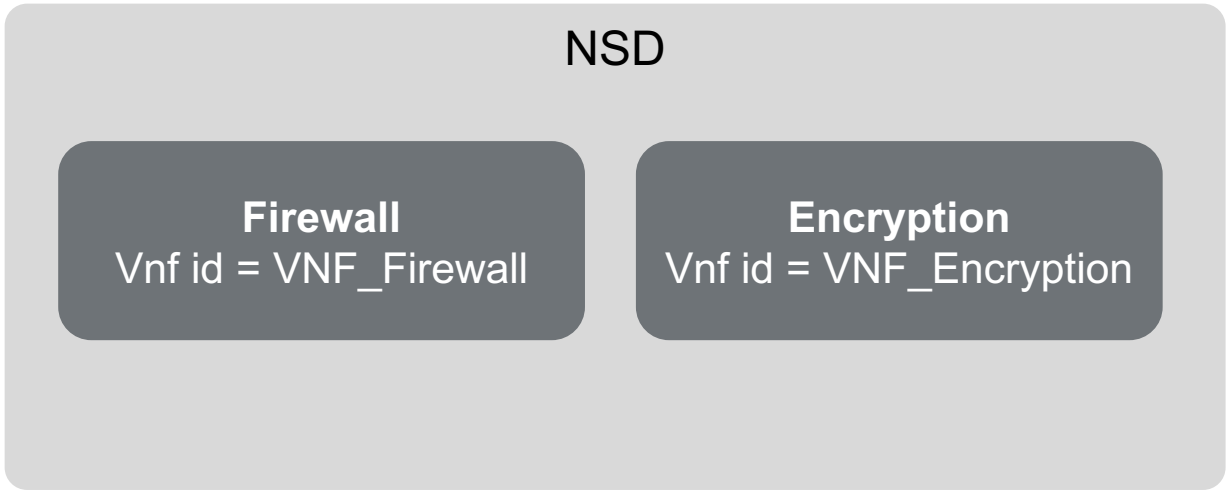
Required fields:

- nsdIdentifier
- designer
- version
- nsdInvariantId

```
"nsdInvariantId": "NSD 2 VNF - ENC and Firewall",
"designer": "Ciena Developer",
"version": "1.0.0",
"nsdIdentifier": "NSD_2_VNF_ENC_FW",
"exportedModelVersion": "0.0.0",
"modelMajorVersion": 1,
```

Required for V2, optional for V1

VNF Descriptor IDs (vnfdId)



Array of IDs for each VNF

IDs must match the *Vnfdid* value of the VNF Design

- NSD can be imported into BP before VNFs are onboarded, but you must onboard the VNFs before onboarding the NSD.

```
"vnfdId": [
      "VNF_ENC", "VNF_Firewall"
    ]
```

Network Service Datapath

Datapath is a combination of a connection points + virtual links + forwarding path

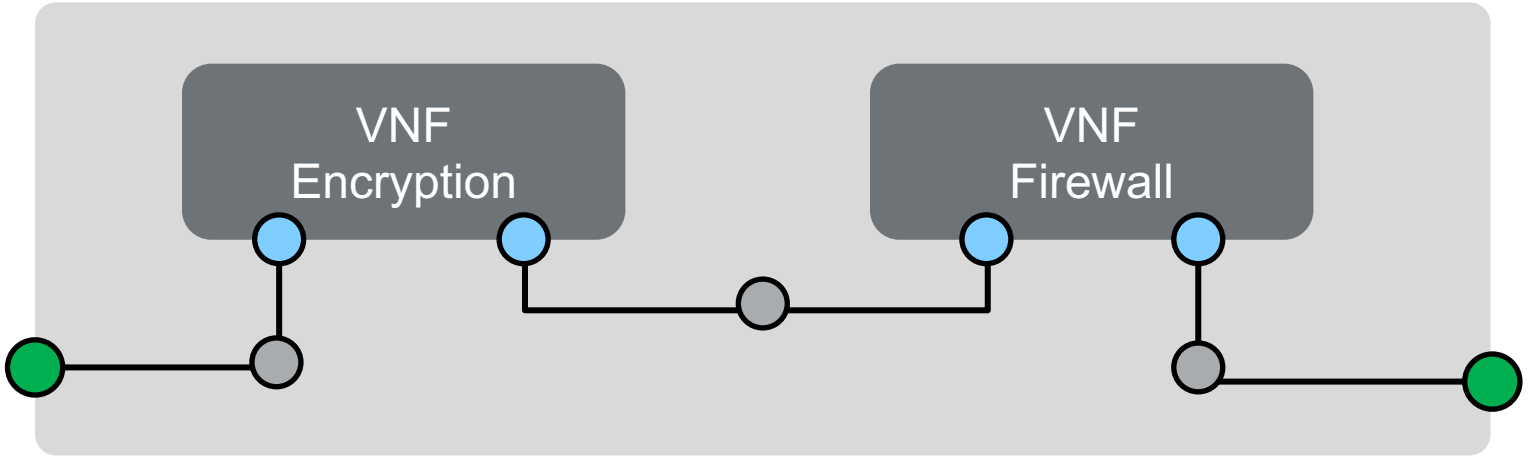
Service access points are defined in *sapd*

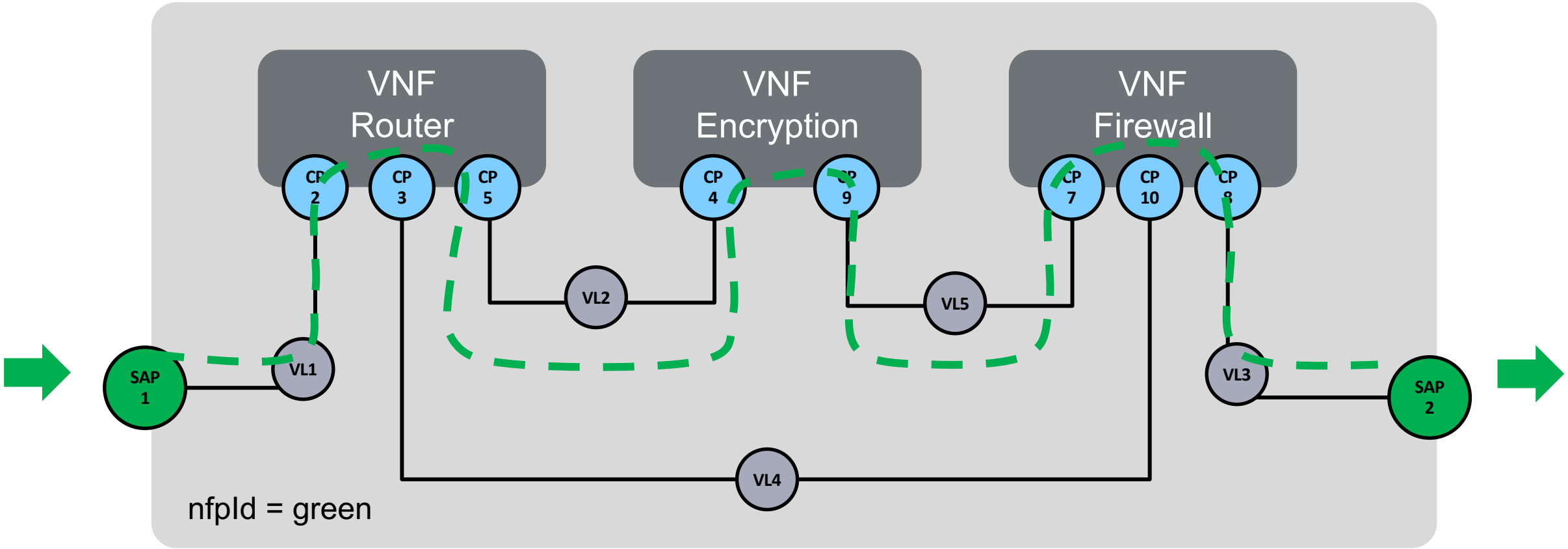
All VNF connection points are listed in *cpdPool*

- cpdId from all VNFD must be included in the cpdPool
- CP pools must include SAP

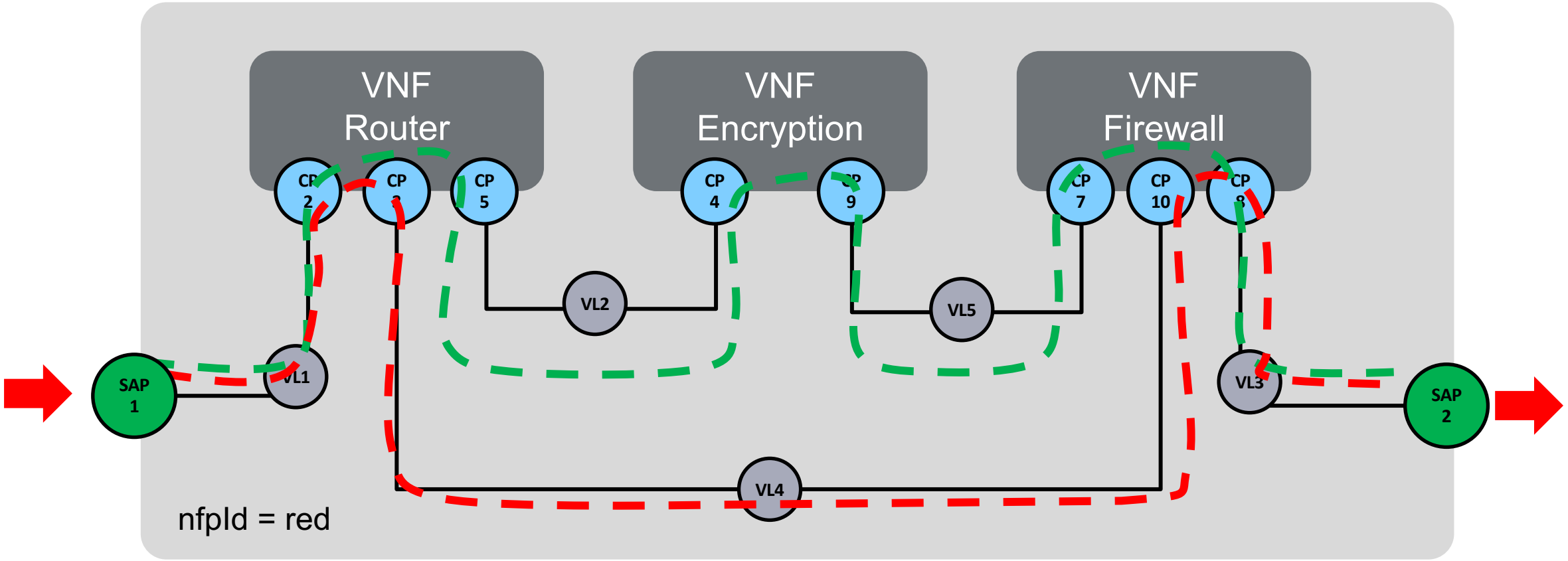
VL properties are defined in *virtualLinkDesc*

Forwarding paths are defined in *vnffgd*

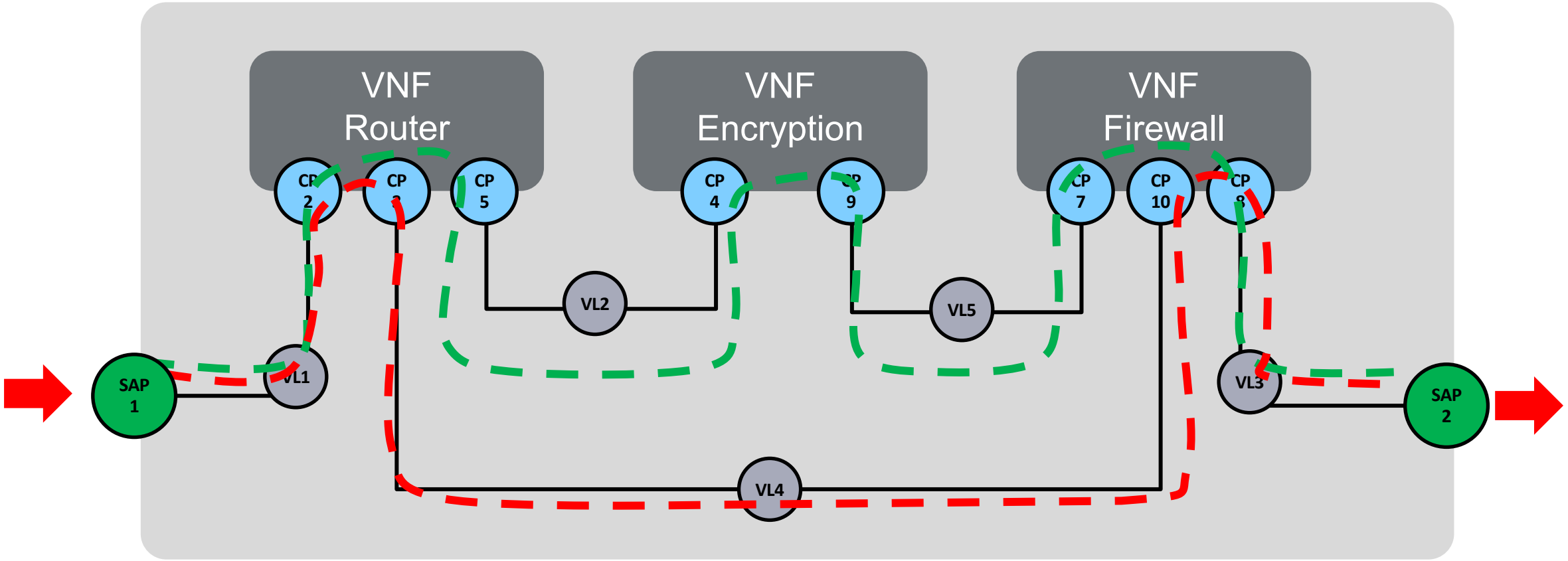




Network Forwarding Path



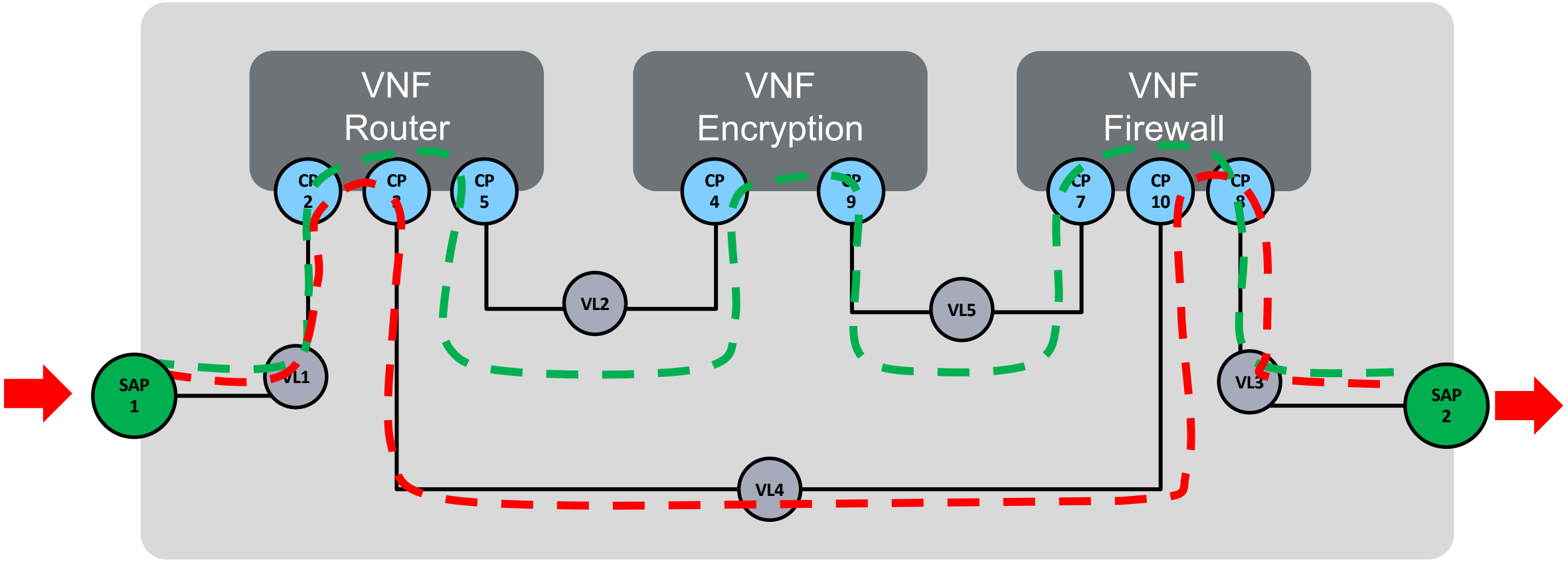
Network Forwarding Path



VNF Forwarding Graph

nfpId = red

nfpId = green



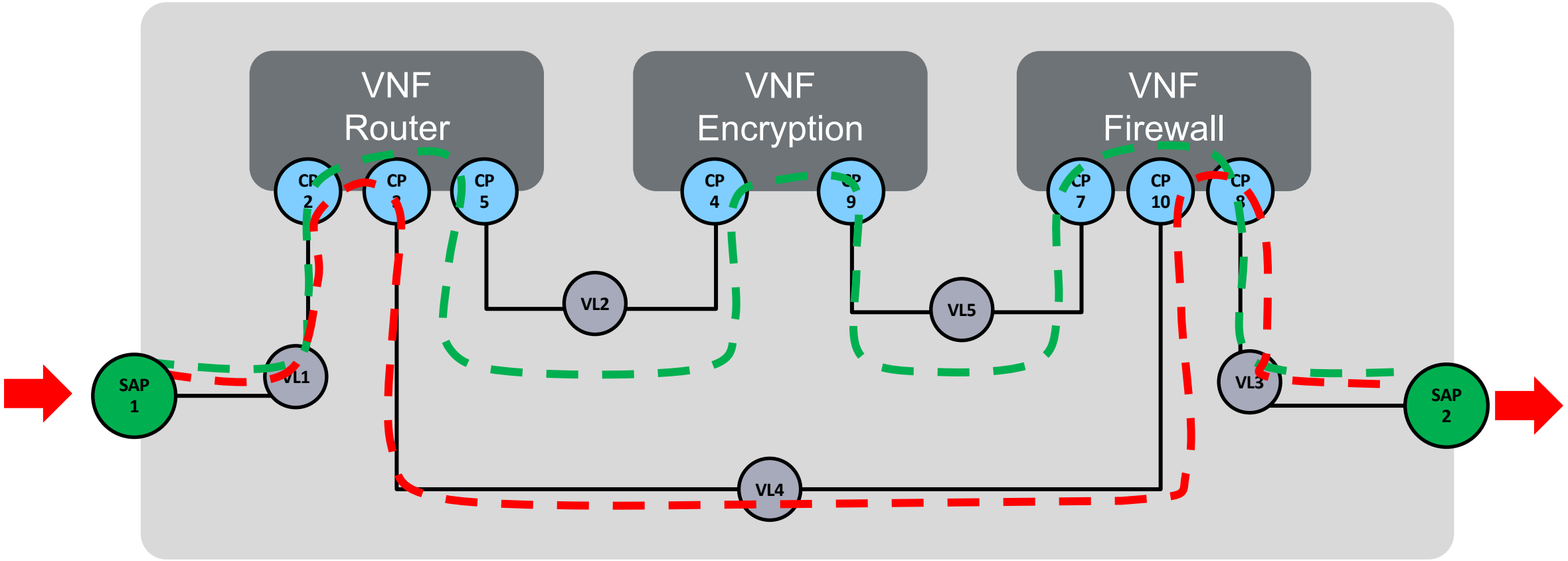
VNF Forwarding Graph

NFP: green, red

VL: VL1, VL2, VL3, VL4, VL5

CP_Pool_1

CP2, CP4, CP5, CP9, CP7, CP10, CP8, SAP1, SAP 2



VNF Forwarding Graph

NFP: green, red

VL: VL1, VL2, VL3, VL4, VL5

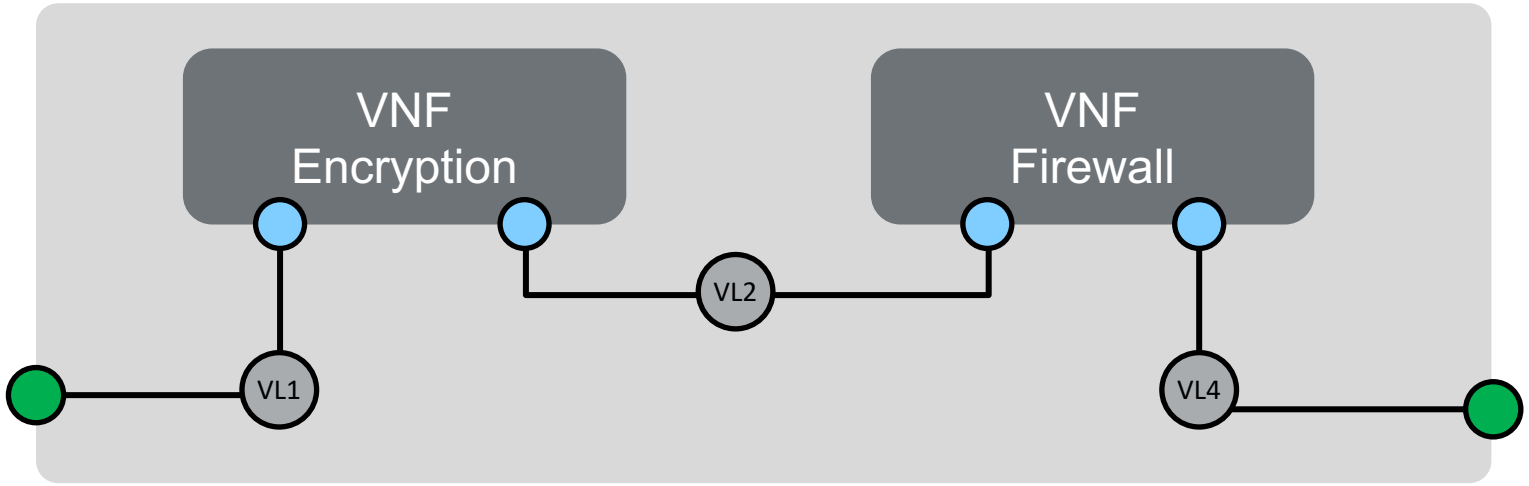
CP Pool: CP_Pool_1

CP_Pool_1

CP2, CP4, CP5, CP9, CP7, CP10, CP8, SAP1, SAP 2

blueplanet
a division of Ciena

Virtual Link Descriptors



Describes connection facilities between VNFs or between VNF and SAP

nfv2.resourceTypes.NsVirtualLinkDesc resource for each descriptor is created during onboarding

Each VL must have unique ID (virtualLinkDescId)

Connectivity type must be specified (connectivityType)

- Ethernet, MPLS, ODU2, IPV4, IPV6, Pseudo-Wire, etc.

Network properties are defined in metadata

Metadata

- **Provides a way to specify VIM specific parameters that are not part of the ETSI specification**
- **Data is passed to VIM during instantiation**
- **VLD and SAP metadata can be included in the NSD**
  - Can be added in UI or JSON
- **Metadata can be added during instantiation**
  - Metadata included during instantiation overrides metadata in NSD
- **See NFVO Developer documentation for a complete list of metadata**

Virtual Link Descriptors Metadata

Title	Resource Type ID	Use
Z VlDesc Key Network Built-In	keyvalue.resourceTypes.ZVlDescKeyVirtualNetworkBuiltin	Specify a built-in network to use
Z VlDesc Key Net OpenStack L2	keyvalue.resourceTypes.ZVlDescKeyVirtualNetworkOpenStackL2	Identify an OpenStack network for the link
Z VlDesc Key Network Subnet	keyvalue.resourceTypes.ZVlDescKeyNetworkSubnet	Identify a subnet for the link
Virtual Network	nfv2.resourceTypes.VirtualNetwork	Network parameters for link

Locations in NSD:

- NsdDesign.virtualLinkDesc.metadata

Can be added/changed during NS instantiation or update

- Not available in instantiation wizard

SAPD Descriptors Metadata

Title	Resource Type ID	Use
Z SAPD Key NetPort FloatIp	keyvalue.resourceTypes.ZSapdKeyVirtualNetworkPortFloatIp	Specify a a floating IP network
Z SAPD Key NetPort Built-In	keyvalue.resourceTypes.ZSapdKeyVirtualNetworkPortBuiltin	Specify a built-in port
Z SAPD Key NetPort Classifiers	keyvalue.resourceTypes.ZSapdKeyVirtualNetworkPortClassifiers	Specify classifiers for the port

Locations in NSD:

- NsdDesign.nsDf.vnfProfile.nsVirtualLinkConnectivity.cpds.metadata
- NsdDesign.sapd.sapd.cpd.metadata

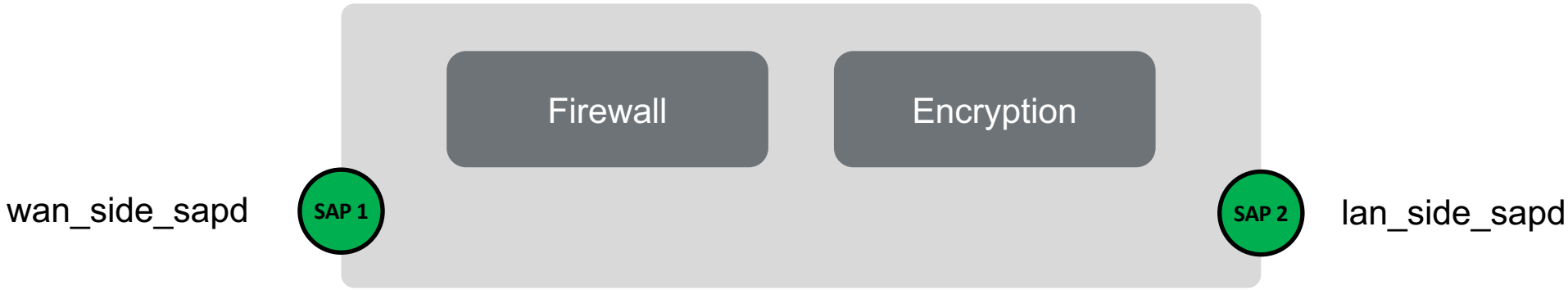
Can be added/changed during NS instantiation or update

- Floating IP is not available in instantiation wizard

Virtual Link Descriptors JSON Example

```
"virtualLinkDesc": [
    {
      "virtualLinkDesc": {
        "virtualLinkDescId": "VL1",
        "connectivityType": {
          "layerProtocol": "IPV4"
        }
      },
      "metadata": {
        "keyValuePairs": [{
          "key": "keyvalue.resourceTypes.ZVlDescKeyNetworkSubnet",
          "value": {
            "networkSubnet": [{
              "prefix": "10.10.10.0/24"
            }]
          }
        }]
      }
    }
 ]
```

Connectivity – Service Access Point Descriptor (sapd)



Defines connectivity in and out of the network service

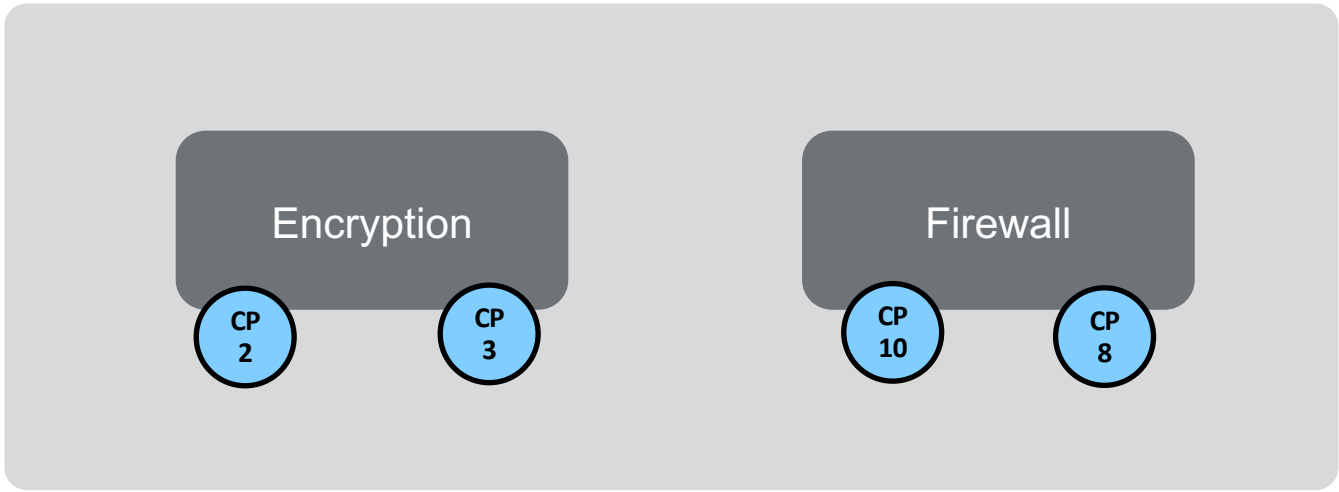
Each SAP must be defined

- SAP is a cp, so all cp properties are valid
- associatedWith: VLD or VNFD
  - VLD – Virtual Link Descriptor defined in the NSD. nsVirtualLinkDescId must specify the ID of the VLD
  - VNFD – associatedCpdId must specify the ID of a vnfExtCpds defined in a VNF Design

Service Access Point JSON Example

```
"sapd": [
      {
        "sapd": {
          "cpd": {
              "cpdId": "CP01”
              "layerProtocol": ["IPv4"],
              "metadata": {
                "keyValuePairs": [{
                     "key": "keyvalue.resourceTypes.ZSapdKeyVirtualNetworkPortBuiltin",
                     "value": {"virtualNetworkPort": { "built-in": "1” } }
                  }
              },
          "nsVirtualLinkDescId": "VL1”,
           "sapAddressAssignment": false
        },
        "associatedWith": "VLD"
      }
]
```

Connectivity – Connection Points



Connection points for VNFs are defined in the VNFD
***cpdId* from VNFD must be used in NSD:**

- cpdPool
- Network Service Deployment Flavor
- Network Service Forwarding Path

CP Pool (cpPool)

List of connection points used in a forwarding graph

- cpdPoolId in forwarding graph identifies the pool to use

Includes sap and VNF cp

```
"cpdPool": [ {
        "cpdPoolId": "Pool1",
        "cpdId": [
          {
            "cpdType": "SAPD",
            "cpdId": "CP01"
          }, {
            "cpdType": "VNFD",
            "cpdId": "CP12",
            "vnfdId": "vnfd_VNF1"
          } ] }
]
```

Specify the *cpType*.

- For SAPD the *cpdId* must match an SAP defined in *sapd*
- For VNF CP, *cpdId* must match the *cpdId* in the VNFD

VNF Forwarding Graph (vnffgd)

Array of forwarding graph objects

Forwarding graphs include:

- nfpd – Array of Network Forwarding Path Descriptors
- vnffgdId – ID of the forwarding graph
- cpdPoolId – Array of the CP Pool IDs. Links the CP Pool to the forwarding path.
- virtualLinkDescId – Array of Virtual Link Descriptor IDs. Links the virtual links to a deployment flavor and virtual link descriptor.
- vnfdId – Array of VNFD IDs. Value must match IDs of onboarded VNFs

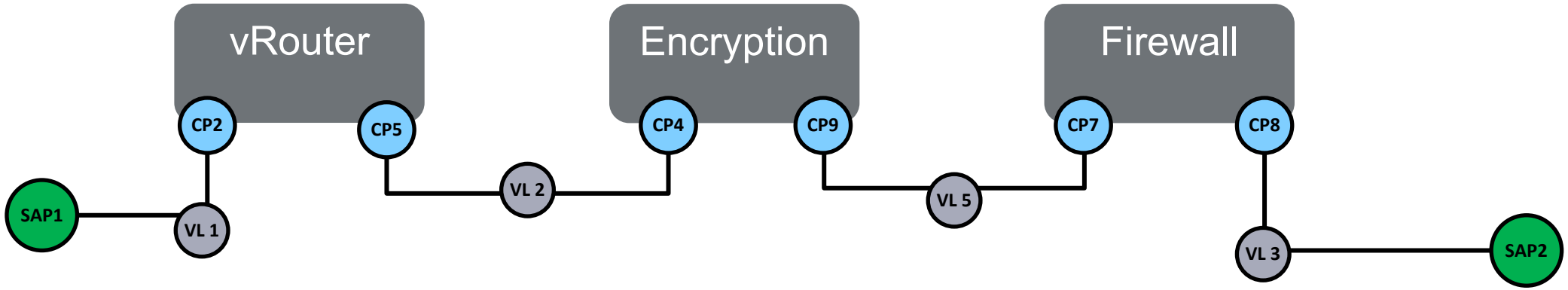
Network Forwarding Path

Ordered list of connection points

Each connection point specifies:

- Connection Point ID
- CP Type

VNF FG can have multiple network forwarding paths




```
CPs: SAP1, CP2, CP5, CP4, CP9, CP7, CP8, SAP2
```

Network Forwarding Path

```
"vnffgd": [ {
        "nfpd": [ {
            "nfpRule": {"name": "nfp_rule" },
            "cpd": [ {
                "cpdType": "SAPD",
                "cpdId": "wan_side"
              }, {
                "cpdType": "VNFD",
                "cpdId": "cp2",
                "vnfdId": "enc_vnf"
              } ],
            "nfpdId": "nnfp_1"
          } ],
        "vnffgdId": "vnf_fwd_graph1",
        "cpdPoolId": [  "cpd_pool_1"  ],
        "virtualLinkDescId": [ "VL1", "VL2", "VL4"  ],
        "vnfdId": ["vnfd_VNF1", "vnfd_VNF2", ]
      } ]
```

Network Forwarding Path
(example does not show an entire NFP)

Network Service Deployment Flavor (nsDf)

Array of deployment flavor objects

- *nsDfId* - Identifier for the flavor
- *virtualLinkProfile* – Array of objects defining bitrate and Virtual Link profile (virtualLinkDesc.virtualLinkDesc.Id)
- *vnfProfile* – One profile for each VNF used in the flavor. Specifies details about the VNF.
- *nsInstantiationLevel* – Array of objects
  - Number of instances of each VNF
  - Mapping of Virtual Link profiles to the level
- *defaultNsInstantiationLevelId* – Required if more there is more than one level
- *flavourKey* – used for closed loop automation. Not implemented in BPO 18.06.

nsDf.virtualLinkProfile

Identifies which links defined in virtualLinkDesc are used in this flavor

```
"virtualLinkProfile": [
      {
         "virtualLinkProfileId": "vl1_profile",
         "virtualLinkDescId": "VL1"
        },
        {
         "virtualLinkProfileId": "vl2_profile",
         "virtualLinkDescId": "VL2"
      }
]
```

nsDf.vnfProfile

instantiationLevel – must match level in VNFD
minNumberOfInstances – integer
maxNumberOfInstances – integer
flavourId – Must match flavor in VNFD
vnfdId – VNFD Id
nsVirtualLinkConnectivity – Array of cpd objects

nsDf.nsInstantiationLevel

Array of objects – one for each instantiation level

nsLevelId – Identifier of this level

vnfToLevelMapping –

- Defines how many VNFs to deploy
- One object for each VNF
- vnfProfileId references nsDf.vnfProfile.vnfProfileId
- Order of VNFs impacts visualization in BP UI

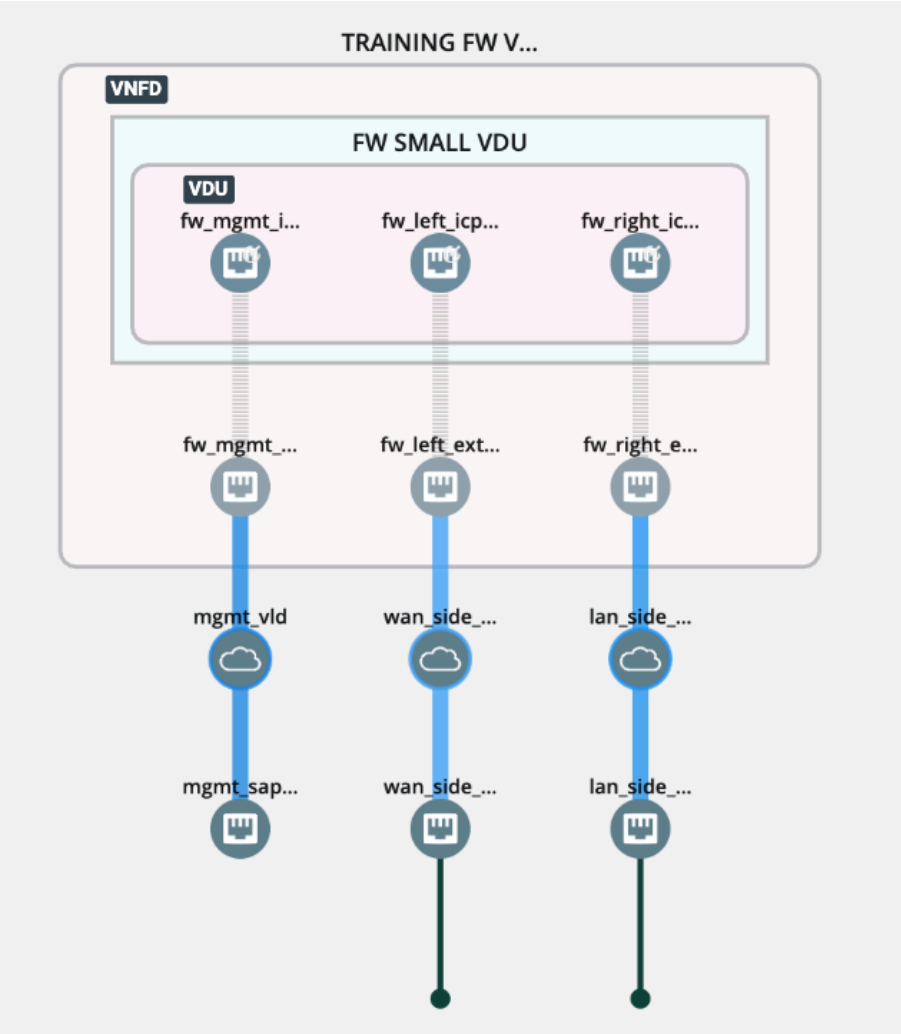
Lab 8: Create an NS Design with NSD Design Builder

Use the UI tool to create an NSD for your firewall VNF

Lab 9: Onboard an NSD

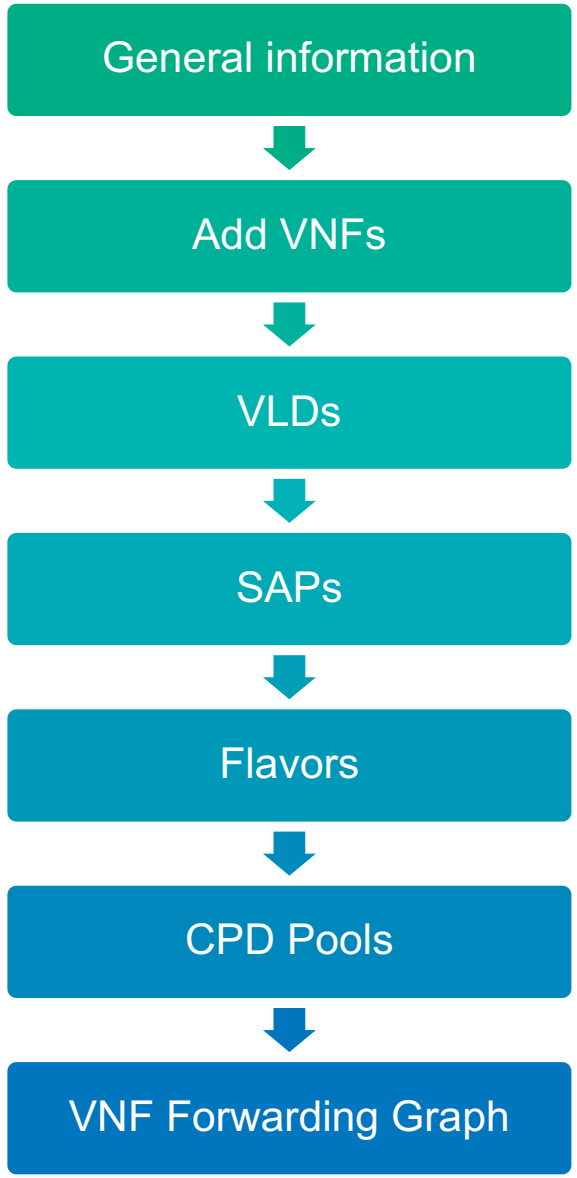
Onboard your NSD design

Instantiate a network service using your design

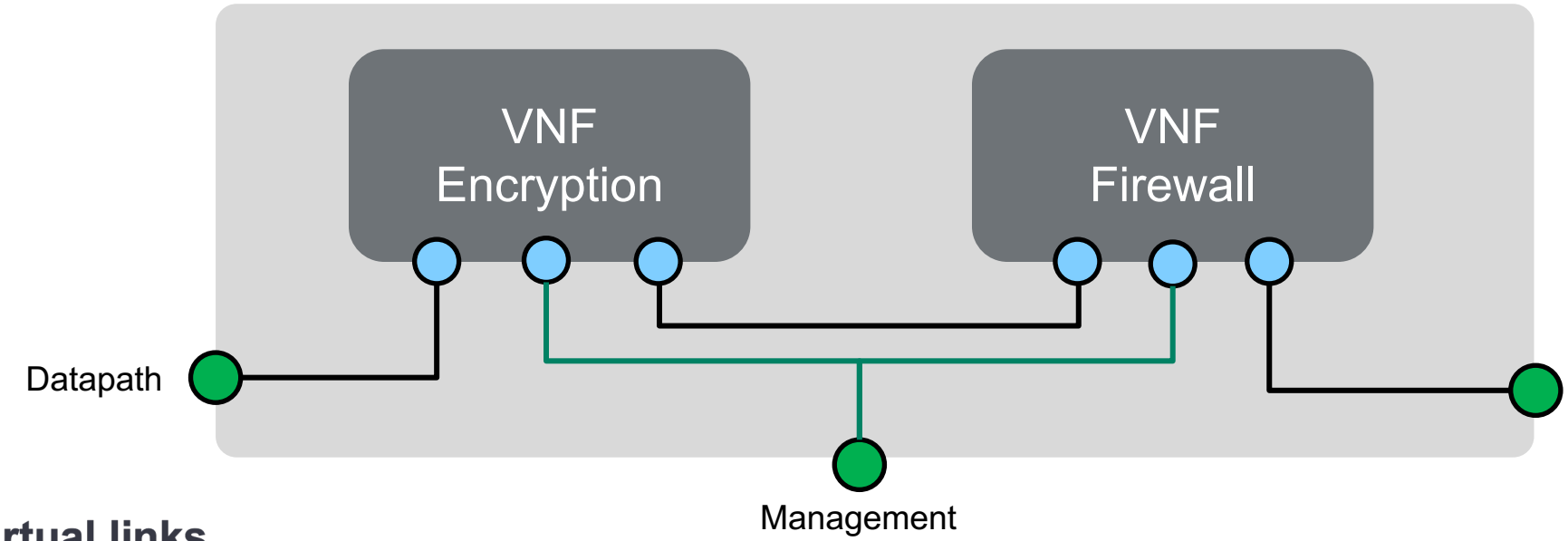


Creating NSD Designs with JSON

- **Useful for larger designs with multiple flavors and levels**
- **Create a new NSD design in Blue Planet**
- **Follow same high-level process as design builder**
- **Use Import operation to import JSON**
  - Must be valid JSON
- **After import, the design can be edited in UI**

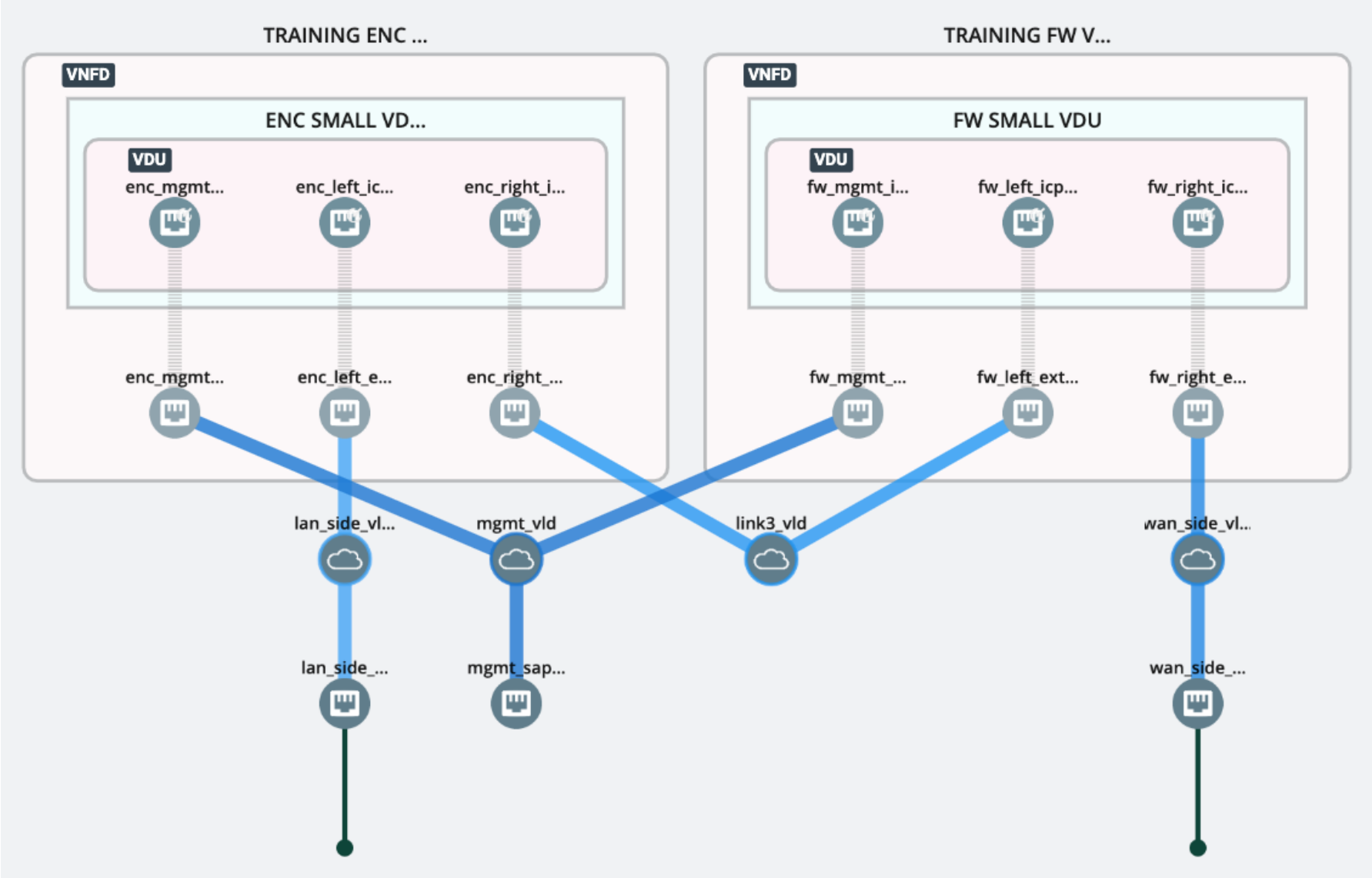


Lab 10: Create an NSD with JSON



- **Four virtual links**
- **Three SAPs**
- **Two flavors:**
  - Small
  - Medium
- **One level**
- **One network forwarding path**

NSD #1 Visualization



Updating Network Services

Updating Existing Network Services

VNF Scaling

- Change VNF level using current VNFD

Change NSD Flavor

- Change service to a different flavor or level within the current NSD.

VNF software modification

- Used to upgrade VNF image to a new version
- Requires new VNFD and new NSD

Associate new NSD version

- Add flavors and levels
- Add new VNFs
- Update VNF software
- Requires new NSD

Update callouts

NSD Requirements for Updating Network Services

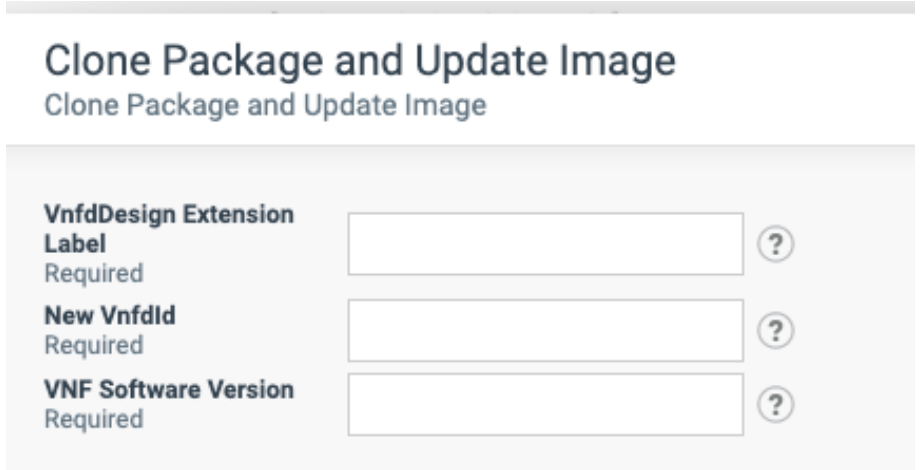
- ***nsdInvariantId* in new NSD must be the same**
- ***version* must be updated**
- **Flavors used by existing services must be in the new NSD, and must not change**

CAUTION:

- Do not edit VNFD or NSD designs that have been onboarded.
- Cloning onboarded descriptors creates a copy of the original design.
- If the NSD or VNFD design has changed, when cloning the NSD the onboarding process might fail.
- Blue Planet assumes original designs have not changed for onboarded descriptors.

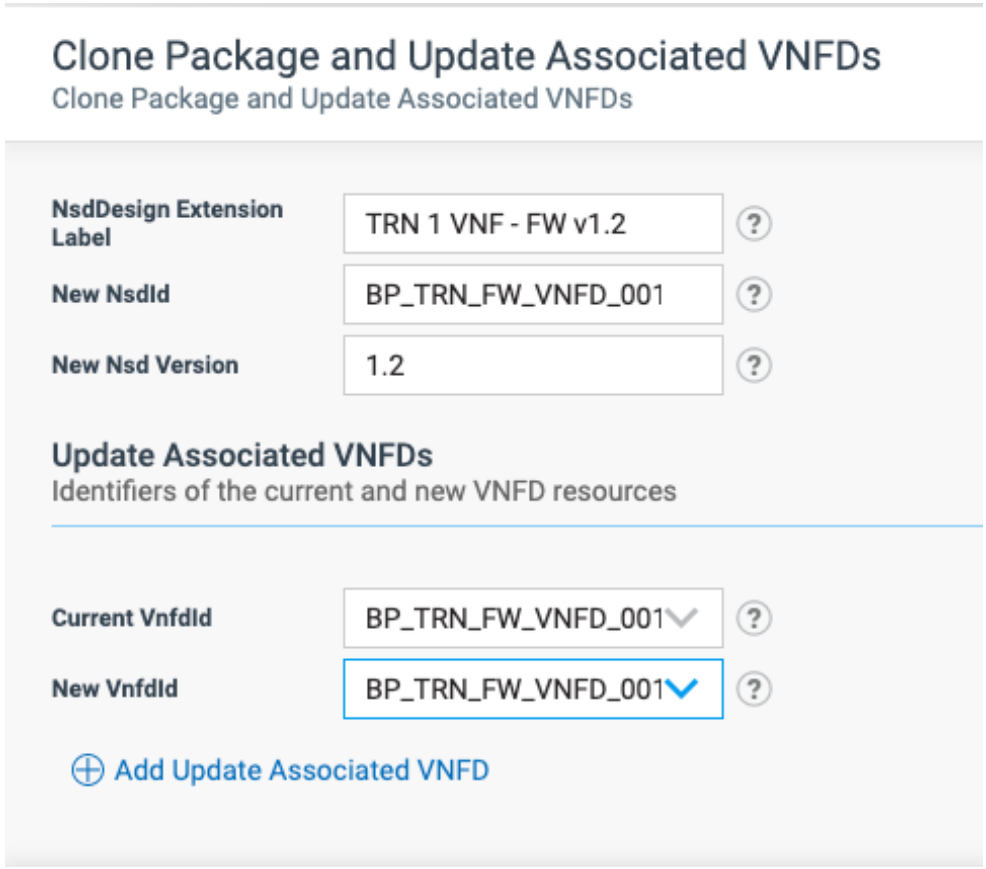
How to Update a VNFD

- **Extend existing VNFD**
  - Creates a new design based on the existing design
  - Use Extend if you want to change the VNFD design
- **Clone Package and Update Image**
  - Use if you only need to update the VNF software image
  - Simplified process for creating and onboarding a new design
  - Requires new image file to be uploaded to SFTP server
  - New VNFD package is ready to be used in an NSD as soon as onboarding completes.



Updating to a New NSD

- **Extend existing NSD**
  - Creates a new design based on the existing design
  - Design can be edited: Add VNFs, add levels, add flavors, etc.
- **Clone Package and Update Image**
  - Creates a copy of the design with no modifications to connectivity or flavors
  - Requires VNFDs to be onboarded
  - New NSD package is ready to be used for services as soon as onboarding completes.
  - Existing network services based on the original descriptor can be updated to the new descriptor



VNF Software Modification

1. **Update VNFD and NSD**
   - Upload new VNF image and MD5 checksum to SFTP server
   - Go to current VNFD Package, run Clone Package and Update Image
   - Go to current NS descriptor, run Clone Package and Update Associated VNFDs
2. **Update network service**
   - Run Update > Associate New NSD Version
     - nsdInvariantId of the current NSD must match the new NSD
   - Run Update > VNF software modification
     - Run from NSInfo details, not the list of services

Lab 11: VNF Software Modification

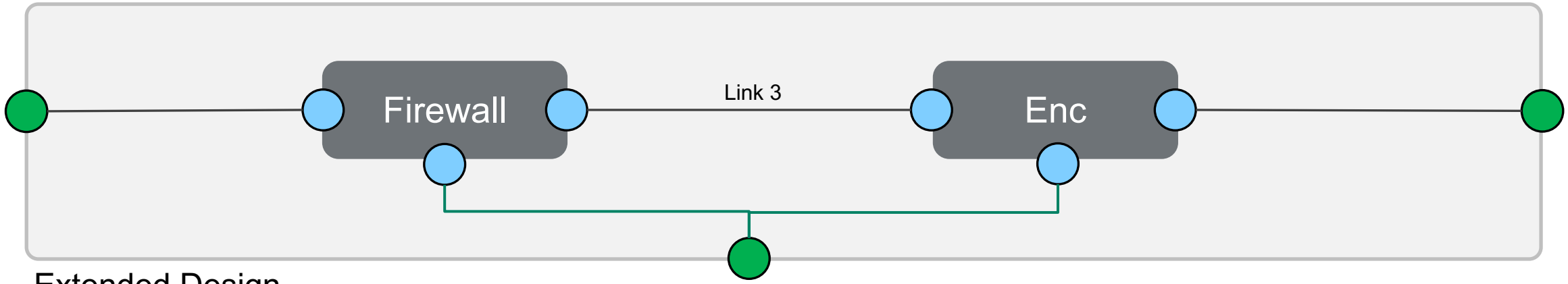
- **Clone VNFD with new software image**
- **Clone NSD**
- **Update your network service**

Extending NSDs

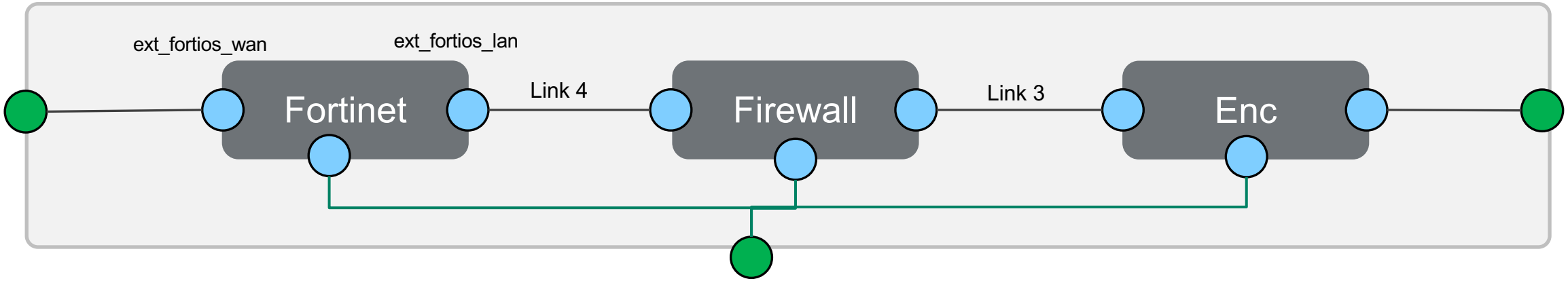
- **Extend operation creates a copy of the NSD design**
- **NSD identifier will be the same as the original, so must be changed.**
- **NSD invariant ID will be the same but does not need to be changed.**
- **Design can be edited in UI, or JSON can be imported.**

Lab 12: Extend an NSD

Current Design



Extended Design

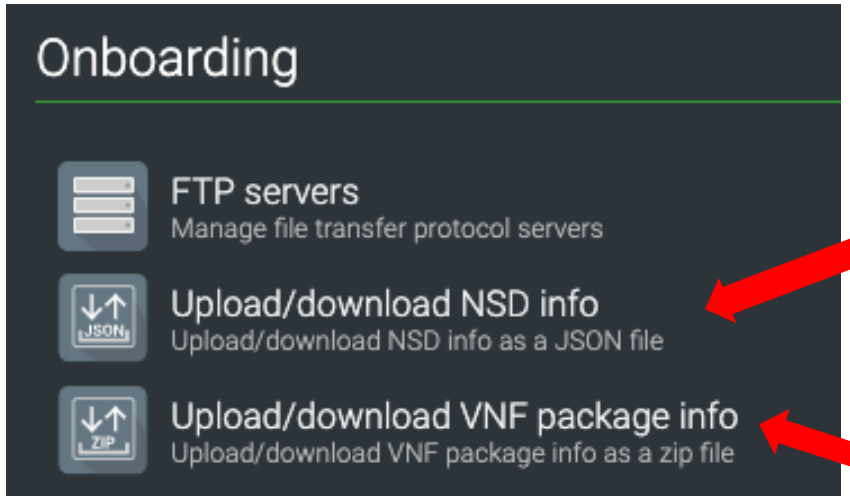


Add a new flavor with three VNFs

Moving Designs

NSD Export

VNFD designs and NSD design can be moved to other Blue Planet servers using the Upload/Download function



NSD – Exports JSON file that can be uploaded to another server

VNFD – Exports CSAR that can be uploaded to another Blue Planet server

Export Operation

- Export writes the contents of a design resource into a JSON description for distribution to other BP systems.

Callouts

Callouts - Overview

- Allows customization of a lifecycle management events by executing a python script plan before or after the main event handling.
- Supported callouts :
  - instantiateNsInterceptor
  - instantiateNsModifier
  - vnfSoftwareModificationInterceptor
  - vnfSoftwareModificationModifier
- Multiple scripts can be run for each event.
  - Callouts have two queues:
    - Interceptor queue – Runs before event handling
    - Modifier queue – Runs after event handling
    - Scripts are prioritized in each queue based on *priority* value
- Callouts run for every network service

Use Case

- **Instantiate NS:**
  - Interceptor: Check to see if VIM has enough resources
    - Instantiation will fail if script returns false
  - Modifier: Send notification of successful instantiation
- **VNF Software Modification:**
  - Interceptor: Backup VNF database before updating software
  - Modifier: Restore backup after software update

- **Create custom script plans**
- **Create and onboard a service template**
  - Implements *callout.resourceTypes.NfvoManoCallout*
  - Sets paths for script plans that will be executed
    - Each script plan is associate with a specific interceptor or modifier callout.
- **Create a callout product**
  - Priority
    - Ciena recommends using a gap in the 5000s between callout priority values. For example, for 3 callouts use priority values 5000, 10000, 15000.
  - Action:
    - continue
    - skipQueue
    - skipMain
  - Max Timeout
    - Default is 30 minutes

Summary

- NSD required for NFVO services
- VNFD required for each VNF
- Flavors and Levels define deployment options
- Traffic flows defined in Network Forwarding Path
- Forwarding Paths are ordered lists of SAPs and CPs
- CPs defined in VNFD
- SAPs defined in NSD

blueplanet®
a division of Ciena

Thank You for Attending