TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0

Committee Specification Draft 01

28 May 2015

Specification URIs

This version:

http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.pdf (Authoritative)
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.html
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.doc

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.pdf (Authoritative)
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.doc

Technical Committee:

OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC

Chairs:

Paul Lipton (paul.lipton@ca.com), CA Technologies
Simon Moser (smoser@de.ibm.com), IBM

Editor:

Shitao Li (lishitao@huawei.com), Huawei Technologies Co., Ltd.

Related work:

This specification is related to:

- *Topology and Orchestration Specification for Cloud Applications Version 1.0*. Edited by Derek Palma and Thomas Spatzier. 25 November 2013. OASIS Standard. Latest version: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html.

Abstract:

The TOSCA NFV profile specifies a NFV specific data model using TOSCA language.

Status:

This document was last revised or approved by the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list, after subscribing to it by following the instructions at the “Send A Comment” button on the TC’s web page at https://www.oasis-open.org/committees/tosca/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/tosca/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[TOSCA-Simple-Profile-NFV-v1.0]

TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0. Edited by Shitao Li. 28 May 2015. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.html. Latest version: http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html.

Notices

Table of Contents

1 Introduction

The TOSCA NFV profile specifies a NFV specific data model using TOSCA language. Network Functions Virtualisation aims to transform the way that network operators architect networks by evolving standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in Datacentres, Network Nodes and in the end user premises.

The deployment and operational behavior requirements of each Network Service in NFV is captured in a deployment template, and stored during the Network Service on-boarding process in a catalogue, for future selection for instantiation. This profile using TOSCA as the deployment template in NFV, and defines the NFV specific types to fulfill the NFV requirements. This profile also gives the general rules when TOSCA used as the deployment template in NFV.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.2 Normative References

[**RFC2119**] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[**ETSI GS NFV-MAN 001 v1.1.1**] Network Functions Virtualisation (NFV); Management and Orchestration

[**TOSCA-1.0**] Topology and Orchestration Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0, an OASIS Standard, 25 November 2013, http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf

[**TOSCA-Simple-Profile-YAML**] TOSCA Simple Profile in YAML Version 1.0

1.3 Non-Normative References

[Reference] [Full reference citation]

(Remove Non-Normative References section if there are none. Remove text below and this note before submitting for publication.)

NOTE: The proper format for citation of technical work produced by an OASIS TC (whether Standards Track or Non-Standards Track) is:

[Citation Label]

Work Product title (italicized). Edited by Albert Alston, Bob Ballston, and Calvin Carlson. Approval date (DD Month YYYY). OASIS Stage Identifier and Revision Number (*e.g.*, OASIS Committee Specification Draft 01). Principal URI (version-specific URI, *e.g*., with stage component: somespec-v1.0-csd01.html). Latest version: (latest version URI, without stage identifiers).

For example:

[OpenDoc-1.2] *Open Document Format for Office Applications (OpenDocument) Version 1.2*. Edited by Patrick Durusau and Michael Brauer. 19 January 2011. OASIS Committee Specification Draft 07. http://docs.oasis-open.org/office/v1.2/csd07/OpenDocument-v1.2-csd07.html. Latest version: http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2.html.

2 Summary of key TOSCA concepts

The TOSCA metamodel uses the concept of service templates to describe cloud workloads as a topology template, which is a graph of node templates modeling the components a workload is made up of and as relationship templates modeling the relations between those components. TOSCA further provides a type system of node types to describe the possible building blocks for constructing a service template, as well as relationship type to describe possible kinds of relations. Both node and relationship types may define lifecycle operations to implement the behavior an orchestration engine can invoke when instantiating a service template. For example, a node type for some software product might provide a 'create' operation to handle the creation of an instance of a component at runtime, or a 'start' or 'stop' operation to handle a start or stop event triggered by an orchestration engine. Those lifecycle operations are backed by implementation artifacts such as scripts or Chef recipes that implement the actual behavior.

An orchestration engine processing a TOSCA service template uses the mentioned lifecycle operations to instantiate single components at runtime, and it uses the relationship between components to derive the order of component instantiation. For example, during the instantiation of a two-tier application that includes a web application that depends on a database, an orchestration engine would first invoke the 'create' operation on the database component to install and configure the database, and it would then invoke the 'create' operation of the web application to install and configure the application (which includes configuration of the database connection).

The TOSCA simple profile assumes a number of base types (node types and relationship types) to be supported by each compliant environment such as a 'Compute' node type, a 'Network' node type or a generic 'Database' node type. Furthermore, it is envisioned that a large number of additional types for use in service templates will be defined by a community over time. Therefore, template authors in many cases will not have to define types themselves but can simply start writing service templates that use existing types. In addition, the simple profile will provide means for easily customizing existing types, for example by providing a customized 'create' script for some software.

3 NFV Overview

Network Functions Virtualization (NFV) leverages standard IT virtualization technology to enable rapid service innovation for Network Operators and Service Providers. Most current networks are comprised of diverse network appliances that are connected—or chained--in a specific way to achieve the desired network service functionality. NFV aims to replace these network appliances with virtualized network functions that can be consolidated onto industry-standard high volume servers, switches and storage, which could be located in data centers, network nodes, or in the end-user premises. These virtual network functions can then be combined using dynamic methods—rather than just static ones—to create and manage network services in an agile fashion.

Deploying and operationalizing end-to-end services in NFV requires software-based tools for Management and Orchestration of virtualized network functions on independently deployed and operated NFV infrastructure platforms. These tools use Network Service Descriptors (NSDs) that capture deployment and operational behavior requirements of each network service. This section describes how NFV models network services using NSDs.

3.1 Network Services

A network service is a composition of Network Functions that defines an end-to-end functional and behavioral specification. Consequently, a network service can be viewed architecturally as a forwarding graph of Network Functions (NFs) interconnected by supporting network infrastructure.

A major change brought by NFV is that virtualization enables dynamic methods rather than just static ones to control how network functions are interconnected and how traffic is routed across those connections between the various network functions.

To enable dynamic composition of network services, NFV introduces Network Service Descriptors (NSDs) that specify the network service to be created. Aside from general information about the service, these Network Service Descriptors typically include two types of graphs:

- A Network Connectivity Topology (NCT) Graph that specifies the Virtual Network Functions that make up the service and the logical connections between virtual network functions. NFV models these logical connections as Virtual Links that need to be created dynamically on top of the physical infrastructure.
- One or more Forwarding Graphs that specify how packets are forwarded between VNFs across the Network Connectivity Topology graph in order to accomplish the desired network service behavior.

A network connectivity topology is only concerned with how the different VNFs are connected, and how data flows across those connections, regardless of the location and placement of the underlying physical network elements. In contrast, the network forwarding graph defines the sequence of VNFs to be traversed by a set of packets matching certain criteria. The network forwarding graph must include the criteria that specify which packets to route through the graph. A simple example of this could be filtering based on a ToS or DSCP value, or routing based on source addresses, or a number of other different applications. Different forwarding graphs could be constructed on the same network connectivity topology based on different matching criteria.

3.2 Network Connectivity Topology

A VNF Network Connectivity Topology (NCT) graph describes how one or more VNFs in a network service are connected to one another, regardless of the location and placement of the underlying physical network elements. A VNF NCT thus defines a logical network-level topology of the VNFs in a graph. Note that the (logical) topology represented by a VNF-NCT may change as a function of changing user requirements, business policies, and/or network context.

In NFV, the properties, relationships, and other metadata of the connections are specified in Virtual Link abstractions. To model how virtual links connect to virtual network functions, NFV introduces uses

Connection Points (CPs) that represent the virtual and/or physical interfaces of the VNFs and their associated properties and other metadata.

The following figure shows a network service example given by the NFV MANO specification [ETSI GS NFV-MAN 001 v1.1.1]. In this example, the network service includes three VNFs. Each VNF exposes different number of connection points.

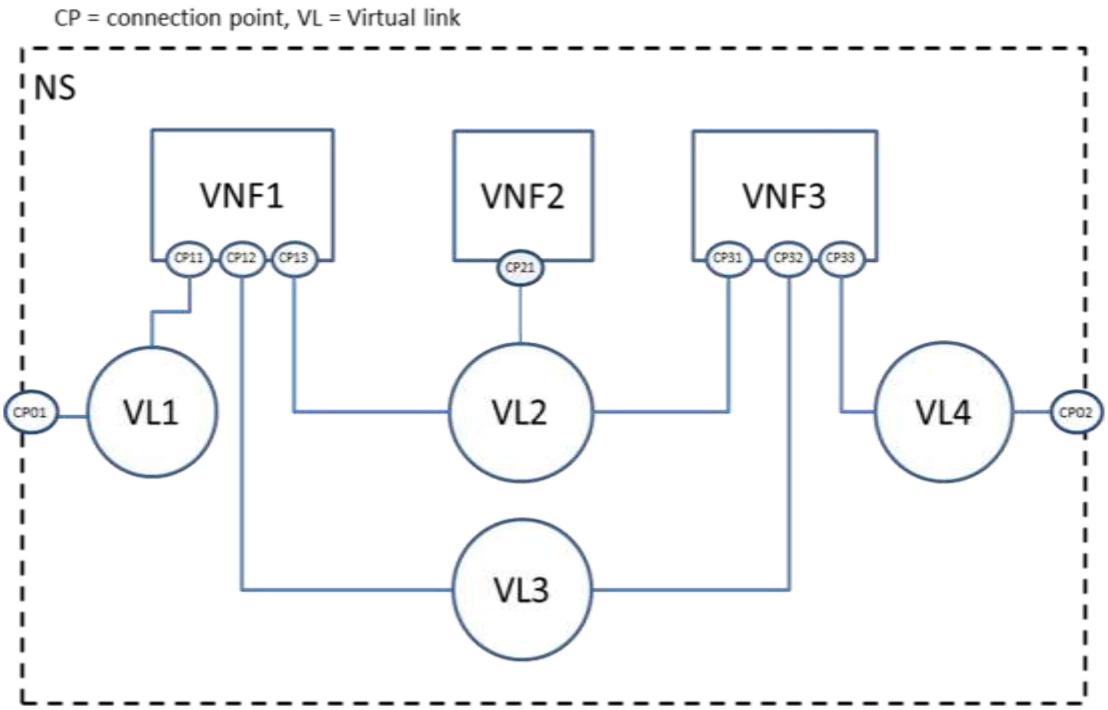

Figure 1. Example network connectivity topology graph

Each Virtual link (VL) describes the basic topology of the connectivity as well as other required parameters (e.g. bandwidth and QoS class). Examples of virtual link types in VNF-NCTs include:

- E-Line, E-LAN, and E-TREE (defined by the Metro Ethernet Forum in MEF Technical Specification MEF 6.1: Ethernet Services Definitions - Phase 2", April, 2008).
- VPLS and VPWS Services (e.g. defined by IETF RFC 4761).
- Different types of Virtual LANs or Private Virtual LANs (e.g. IETF RFC 3069).
- Different types of Layer 2 Virtual Private Networks (e.g. IETF RFC 4464).
- Different types of Layer 3 Virtual Private Networks (e.g. IETF RFC 3809).
- Different types of Multi-Protocol Label Switching Networks (e.g. IETF RFC 3031).
- Other types of layer 2 services, such as Pseudo Wire Switching for providing multiple Virtual Leased Line Services (e.g. IETF RFC 4385).

3.3 Network Forwarding Graph

A VNF forwarding graph is specified by a Network Service Provider to define how traffic matching certain criteria is intended to flow through one or more network functions in a Network Connectivity Topology in order to accomplish the desired network service functionality. The NFV specification describes network forwarding graphs using one or more Network Forwarding Paths. A Network Forwarding Path is an ordered lists of Connection Points that form a chain of VNFs. The order of network functions applied is application-dependent, and may be a simple sequential set of functions, or a more complex graph with alternative paths (e.g. the service may fork, and even later combine), depending on the nature of the traffic, the context of the network, and other factors.

The following figure shows an example of two VNF Forwarding Graphs established on top of the Network Connectivity Topology described earlier. VNFFG1 has two Network Forwarding Paths (VNFFG1:NFP1 and VNFFG1:NFP2) whereas VNFFG2 only has a single NFP (VNFFG2:NFP1).

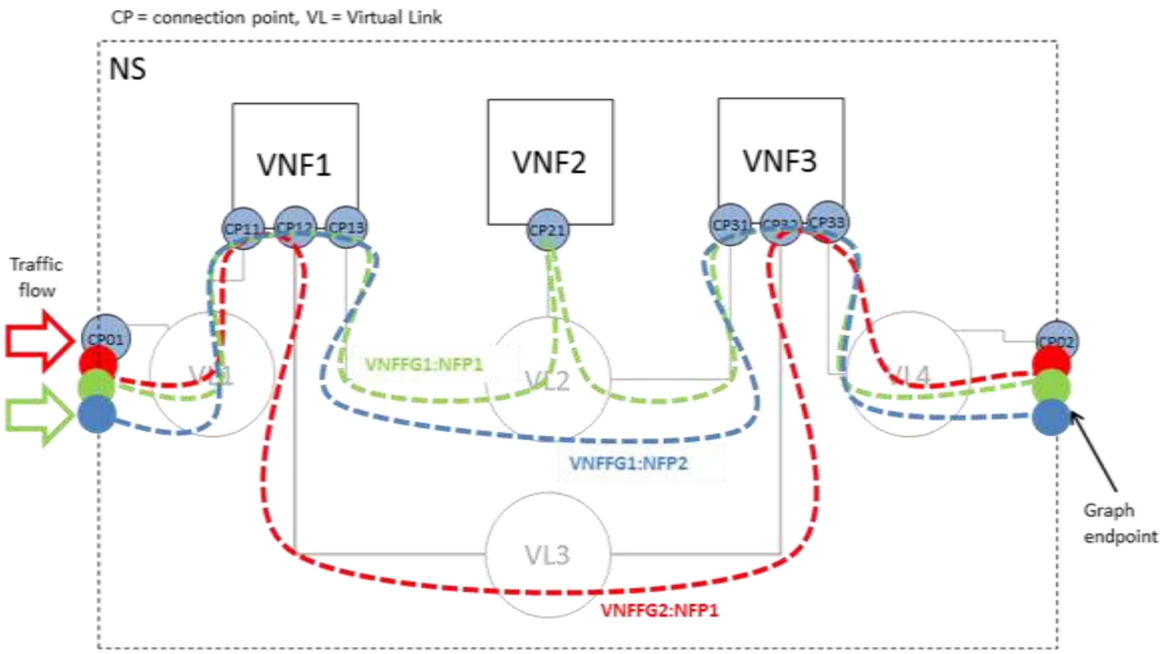

Figure 2. Multiple forwarding graphs using the same network connectivity graph

4 Deployment Template in NFV

The deployment template in NFV fully describes the attributes and requirements necessary to realize such a Network Service. Network Service Orchestration coordinates the lifecycle of VNFs that jointly realize a Network Service. This includes (not limited to) managing the associations between different VNFs, the topology of the Network Service, and the VNFFGs associated with the Network Service.

The deployment template for a network service in NFV is called a network service descriptor (NSD), it describes a relationship between VNFs and possibly PNFs that it contains and the links needed to connect VNFs.

There are four information elements defined apart from the top level Network Service (NS) information element:

- Virtualized Network Function (VNF) information element
- Physical Network Function (PNF) information element
- Virtual Link (VL) information element
- VNF Forwarding Graph (VNFFG) information element

A VNF Descriptor (VNFD) is a deployment template which describes a VNF in terms of its deployment and operational behavior requirements.

A VNF Forwarding Graph Descriptor (VNFFGD) is a deployment template which describes a topology of the Network Service or a portion of the Network Service, by referencing VNFs and PNFs and Virtual Links that connect them.

A Virtual Link Descriptor (VLD) is a deployment template which describes the resource requirements that are needed for a link between VNFs, PNFs and endpoints of the Network Service, which could be met by various link options that are available in the NFVI.

A Physical Network Function Descriptor (PNFD) describes the connectivity, Interface and KPIs requirements of Virtual Links to an attached Physical Network Function.

The NFVO receives all descriptors and on-boards to the catalogues, NSD, VNFFGD, and VLD are "on-boarded" into a NS Catalogue; VNFD is on-boarded in a VNF Catalogue, as part of a VNF Package. At the instantiation procedure, the sender (operator) sends an instantiation request which contains instantiation input parameters that are used to customize a specific instantiation of a network service or VNF. Instantiation input parameters contain information that identifies a deployment flavor to be used and those parameters used for the specific instance.

5 General Mapping between TOSCA and NFV Deployment Template

At the top level of TOSCA data model is a service template, within a service template, it includes several node templates with different types. In NFV, NSD is at the top level, under NSD, it includes VNFD, VNFFGD, VLD and PNFD. The mapping between TOSCA and NFV takes the following approach.

1. NSD is described by using a service template,
2. VNFD, VNFFGD, VLD and PNFD is considered as node templates with appropriate node types.
3. VNFD can be further described by using another service template with substitutable node type.

The mapping relationship between TOSCA and NFV is showing in Figure 3.

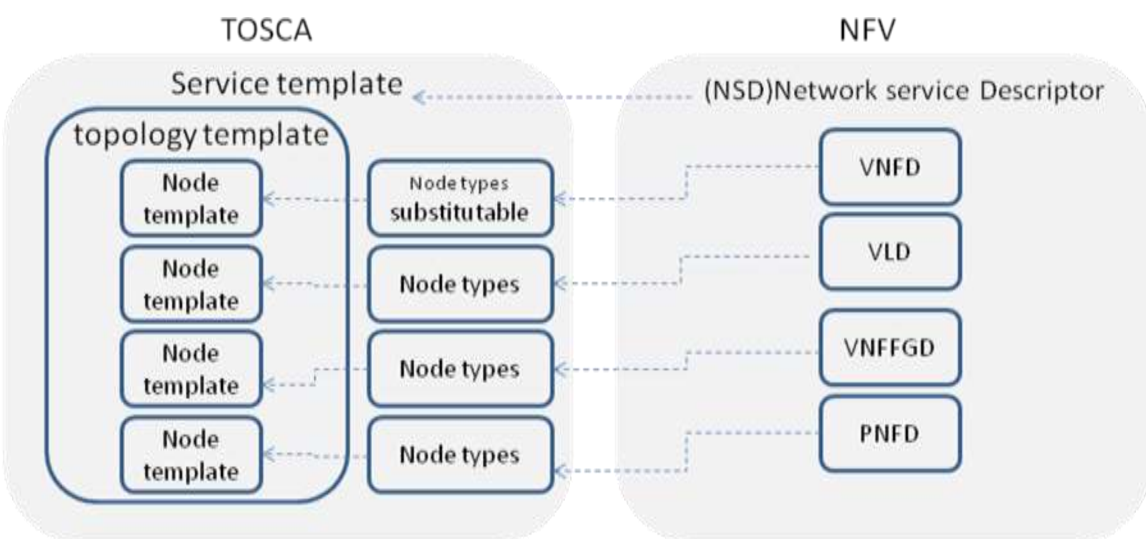

Figure 3. General mapping between TOSCA and NFV

6 TOSCA Data Model for NSD

As described in NFV, NSD describes the attributes and requirements necessary to realize a Network Service. Figure 2 is a network service example given by NFV MANO specification [ETSI GS NFV-MAN 001 v1.1.1]. In this example, the network service includes three VNFs. Each VNF exposes different number of connection points, which represent the virtual and/or physical interface of VNFs. Virtual link (VL) describes the basic topology of the connectivity (e.g. ELAN, ELINE, ETREE) between one or more VNFs connected to this VL and other required parameters (e.g. bandwidth and QoS class).

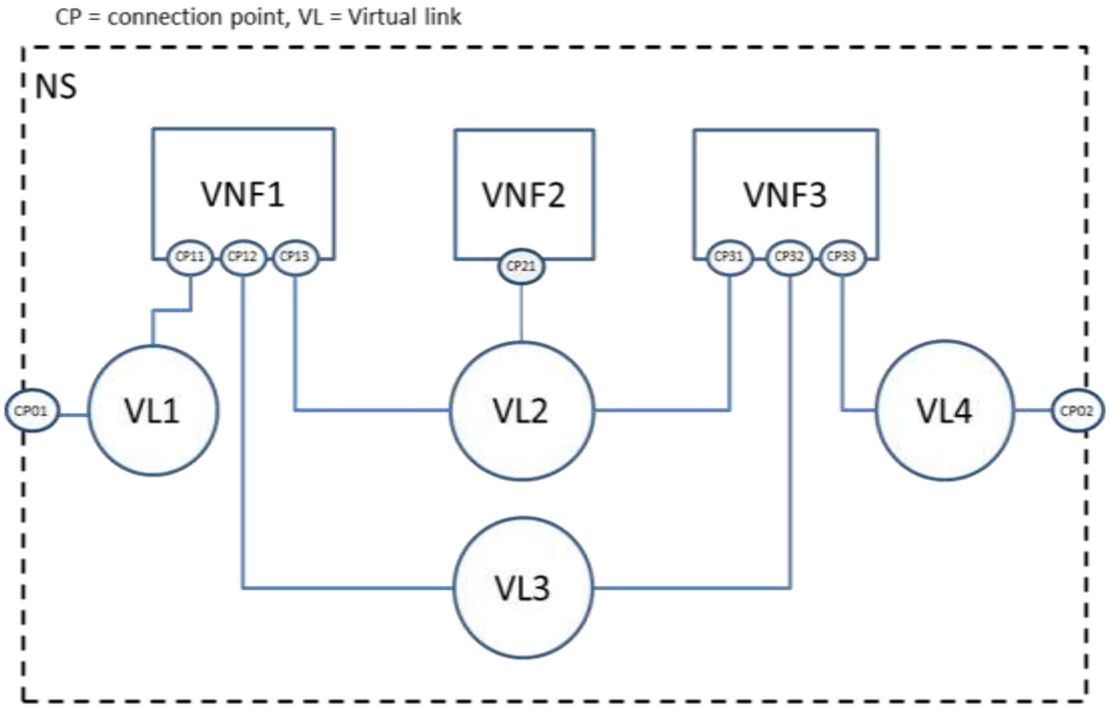

Figure 4. Network service example for NFV

For simplicity, the VNF and its connection point can be considered as a subsystem of the network service. And a new relationship type is needed to connect VNF and virtual link. Figure 3 shows how the TOSCA node, capability and relationship types enable modeling the NFV application using virtualLinkTo relationship between VNF and virtual link.

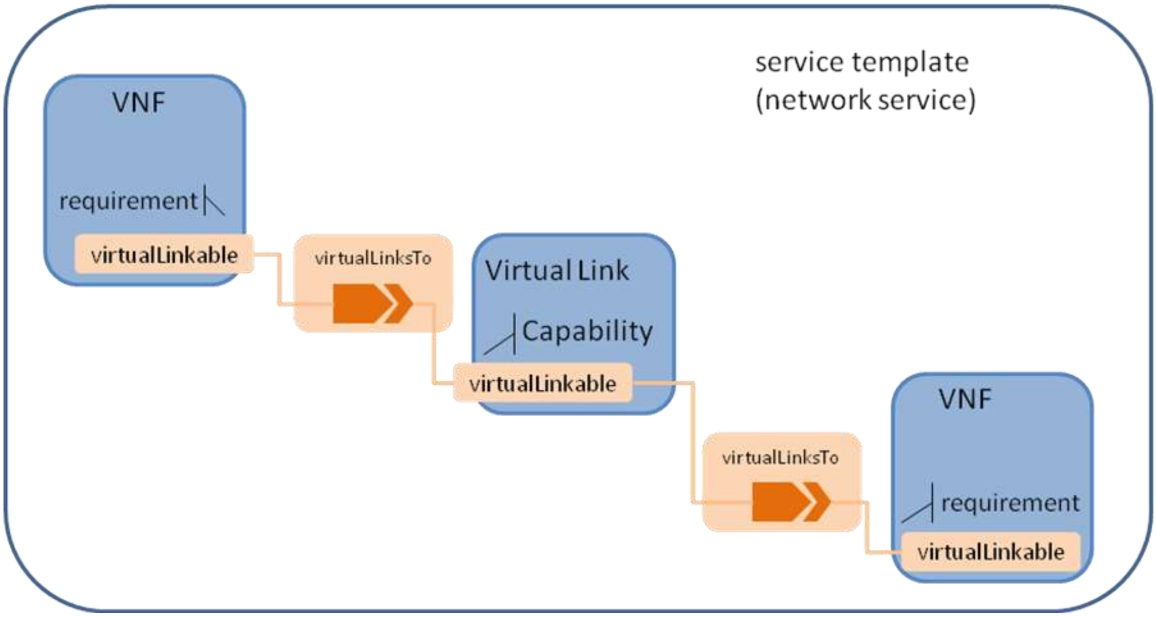

Figure 5. TOSCA node, capability and relationship types used in NFV application

The virtualLinkable requirement of VNF is exposed by the connection point of that VNF who act as an endpoint.

6.1 Namespace and Alias

The following table defines the namespace alias and (target) namespace values that SHALL be used when referencing the TOSCA simple Profile for NFV version 1.0.0 specification.

Alias	Target Namespace	Specification Description
tosca_simple_profile_for_nfv_1_0_0	http://docs.oasis-open.org/tosca/ns/simple/yaml/1.0/nfv/1.0/	The TOSCA Simple Profile for NFV v1.0.0 target namespace and namespace alias.

6.2 Using service template for a NFV network service

The use case of a network service is shown in Figure 6. This section uses a TOSCA service template to describe the network service as shown in Figure 4.

```
tosca_definitions_version:      tosca_simple_profile_for_nfv_1_0_0
tosca_default_namespace:    # Optional. default namespace (schema, types version)
template_name:                      # Optional name of this service template
template_author:                     # Optional author of this service template
template_version:                   # Optional version of this service template
description: example for a NSD.
service_properties:
    ID:                                     # ID of this Network Service Descriptor
    vendor:                             # Provider or vendor of the Network Service
    version:                             # Version of the Network Service Descriptor
imports:
    - tosca_base_type_definition.yaml
    # list of import statements for importing other definitions files
```

```
topology_template:
    inputs:
            flavor ID:
     VNF1:
            type: tosca.nodes.nfv.VNF.VNF1
            properties:
                Scaling_methodology:
                Flavour_ID:
                Threshold:
                Auto-scale policy value:
                Constraints:
            requirements:
               virtualLink: VL1
               virtualLink: VL2
               virtualLink: VL3

     VNF2:
            type: tosca.nodes.nfv.VNF.VNF2
            properties:
                Scaling_methodology:
                Flavour_ID:
                Threshold:
                Auto-scale policy value:
                Constraints:
            requirements:
               virtualLink: VL2

     VNF3:
            type: tosca.nodes.nfv.VNF.VNF3
            properties:
                Scaling_methodology:
                Flavour_ID:
                Threshold:
                Auto-scale policy value:
                Constraints:
            requirements:
               virtualLink: VL2
               virtualLink: VL3
               virtualLink: VL4

      CP01          #endpoints of NS
            type: tosca.nodes.nfv.CP
```



```
        properties:
            type:
        requirements:
            virtualLink: VL1


  CP02        #endpoints of NS
        type: tosca.nodes.nfv.CP
        properties:
            type:
        requirements:
            virtualLink: VL4

  VL1
        type: tosca.nodes.nfv.VL.Eline
        properties:
            # omitted here for brevity
        capabilities:
            -virtual_linkable
              occurrences: 2

  VL2
        type: tosca.nodes.nfv.VL.ELAN
        properties:
            # omitted here for brevity
        capabilities:
            -virtual_linkable
              occurrences: 5
  VL3
        type: tosca.nodes.nfv.VL.Eline
        properties:
            # omitted here for brevity
        capabilities:
            -virtual_linkable
              occurrences: 2


  VL4
        type: tosca.nodes.nfv.VL.Eline
        properties:
            # omitted here for brevity
         capabilities:
```



```
            -virtual_linkable
              occurrences: 2
    VNFFG:
```

Figure 6.   TOSCA template for NSD

In the example above, service_properties is used to define service specific properties, such as ID, vender, version. Each VNF is described as a node template.VNF1 has three virtualLinkable requirements, each for a different virtual link, VL1, VL2 and VL3. VNF2 only has virtualLinkable requirement to VL2. VNF3 has three virtualLinkable requirements to VL2, VL3, VL4 respectively. CP01 and CP02 are acting as the endpoint of the network service, they are both described as node templates with port node type as defined in **[TOSCA-Simple-Profile-YAML]**. CP01 has virtualLinkable requirement to VL1, and CP02 has virtualLinkable requirement to VL4. VL1, VL2, VL3 and VL4 are described as node templates with tosca.nodes.nfv.virtualLink node type.

6.3 Capability types

6.3.1 tosca.capabilities.nfv.VirtualLinkable

A node type that includes the VirtualLinkable capability indicates that it can be pointed by tosca.relationships.nfv.VirtualLinksTo relationship type.

Shorthand Name	VirtualLinkable
Type Qualified Name	tosca:VirtualLinkable
Type URI	tosca.capabilities.nfv.VirtualLinkable

6.3.1.1 Properties

Name	Required	Type	Constraints	Description
N/A	N/A	N/A	N/A	N/A

6.3.1.2 Definition

```
tosca.capabilities.nfv.VirtualLinkable:
  derived_from: tosca.capabilities.Root
```

6.4 Relationship Types

6.4.1 tosca.relationships.nfv.VirtualLinksTo

This relationship type represents an association relationship between VNFs and VL node types.

Shorthand Name	VirtualLinksTo
Type Qualified Name	tosca:VirtualLinksTo
Type URI	tosca.relationships.nfv.VirtualLinksTo

6.4.1.1 Definition

```
tosca.relationships.nfv.VirtualLinksTo:
  derived_from: tosca.relationships.ConnectsTo
  valid_target_types: [ tosca.capabilities.nfv.VirtualLinkable ]
```


7 TOSCA Data Model for VNFD

A VNF can be considered as a subsystem in a network service, it can include:

- VDU, which is a subset of a VNF. A VDU can be mapped to a single VM;
- Connection point, some of connection points are only used to connect internal virtual link, while others are exposed to connect outside virtual link. A connection point has to bind with a VDU.
- Internal virtual link, the main functionalities are the same with the virtual link defined in the network service level, but it is only used within VNF to provide connectivity between VDUs.

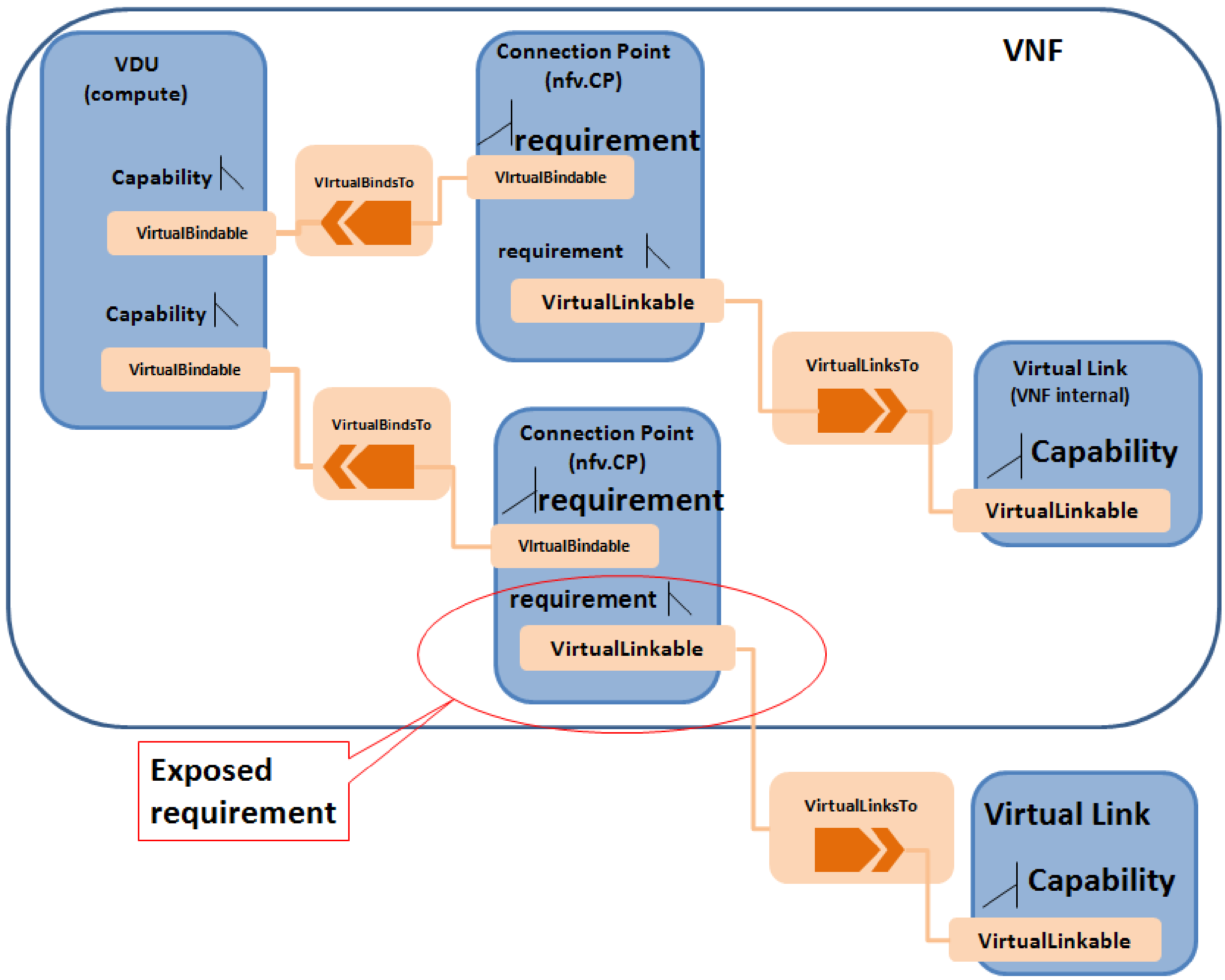

Figure 7. TOSCA node, capability and relationship types used in VNF application

7.1 Node Template Substitution Mapping for a VNF

The substitution mapping feature as defined in [TOSCA-Simple-Profile-YAML], is used to define a new node type, which its characteristics can be mapped to internal elements of a service template.

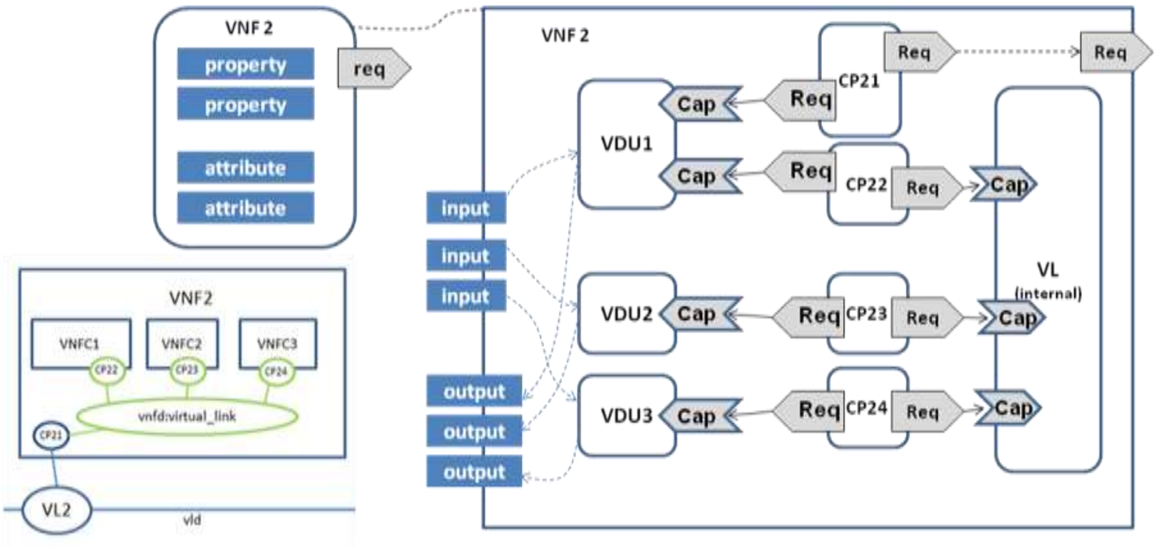

Figure 8. *Substitution mapping for a VNF node type to a service template*

Figure 8 shows an example of the internal structure of a VNF. In this example, VNF2 comprises 3 VDUs which connect to an internal Virtual Link. The first VDU has two Connection Points: one (CP21) to the external Virtual Link, another one (CP22) to the internal Virtual Link. VDU provides the capability Bindable to bind connection point. Connection point has two requirements, one to VDU with bindable, another to virtual link with virtualLinkable. The connection point that has the requirement to the external virtual link exposes the virtualLinkable requirement of the VNF. In the example as shown in Figure 8, CP21 exposes the virtualLinkable requirement of VNF2.

```
tosca_definitions _version:      tosca_simple_profile_for_nfv_1_0_0
tosca_default_namespace :    # Optional. default namespace (schema, types version)
template_name:                     # Optional name of this service template
template_author:                    # Optional author of this service template
template_version:                  # Optional version of this service template
description: example for VNF2
service_properties:
    ID:                                     # ID of this VNF Descriptor
    vendor:                             # Provider or vendor of the VNF
    version:                             # Version of VNF software, described by the
descriptor under consideration
imports:
    - tosca_base_type_definition.yaml
  # list of import statements for importing other definitions files
topology_template:

inputs:

subsititution_mappings:
     node_type: tosca.nodes.nfv.VNF.VNF2
     requirements:
```



```
        virtualLinkable: [CP21, virtualLinkable]

node_templates:
     VDU1:
        type: tosca.nodes.nfv.VDU
        properties:
           # omitted here for brivity
        requirements:
           - host:
             node_filter:
               capabilities:
                 # Constraints for selecting “host” (Container Capability)
                 - host
                    properties:
                      - num_cpus: { in_range: [ 1, 4 ] }
                      - mem_size: { greater_or_equal: 2 GB }
                 # Constraints for selecting “os” (OperatingSystem Capability)
                 - os:
                    properties:
                      - architecture: { equal: x86_64 }
                      - type: linux
                      - distribution: ubuntu       Interfaces:
                       # omitted here for brevity
        artifacts:
           VM_image:vdu1.image  #the VM image of VDU1
        Interface:
          Standard:
          create:vdu1_install.sh
          configure:
          implementation: vdu1_configure.sh

     VDU2:
        type: tosca.nodes.nfv.VDU
        properties:
          # omitted here for brivity

     VDU3:
        type: tosca.nodes.nfv.VDU
        properties:
          # omitted here for brivity

      CP21:          #endpoints of VNF2
```



```
      type: tosca.nodes.nfv.CP
      properties:
        type:
      requirements:
        virtualbinding: VDU1

   CP22:
      type: tosca.nodes.nfv.CP
      properties:
        type:
      requirements:
        virtualbinding: VDU1
        virtualLink: internal_VL

   CP23
      type: tosca.nodes.nfv.CP
      properties:
        type:
      requirements:
        virtualbinding: VDU2
        virtualLink: internal_VL

   CP24
      type: tosca.nodes.nfv.CP
      properties:
        type:
      requirements:
        virtualbinding: VDU3
        virtualLink: internal_VL


    internal_VL
      type: tosca.nodes.nfv.VL.ELAN
      properties:
        # omitted here for brivity
      capabilities:
        -virtual_linkable
           occurrences: 5
```

Figure 9. TOSCA template for VNFD

In the example above, ID, vender and version are defined service_properties for VNFD specific usage. The topology_template defines the internal structure of VNF2. In the subsititution_mappings element, it defines the node type as tosca.nodes.nfv.vnf2 which is the substitutable node type as defined by this service template. The virtualLinkable requirement is exposed by the virtualLinkable requirement of CP21.

VDU as a compute component in VNF, has requirement for compute and memory, it may also include VM image, which can be described as artifact. CP21 as the endpoint of VNF2, has binding requirement for VDU1, and virtualLinkable requirement for external virtual link. CP22, CP23 and CP24 are internal connection point of VNF2, which all connect to the internal_VL.

7.2 Capability Types

7.2.1 tosca.capabilites.nfv.VirtualBindable

A node type that includes the VirtualBindable capability indicates that it can be pointed by tosca.relationships.nfv.VirtualBindsTo relationship type.

Shorthand Name	VirtualBindable
Type Qualified Name	tosca: VirtualBindable
Type URI	tosca.capabilities.nfv.VirtualBindable

7.2.1.1 Properties

Name	Required	Type	Constraints	Description
N/A	N/A	N/A	N/A	N/A

7.2.1.2 Definition

```
tosca.capabilities.nfv.VirtualBindable:
  derived_from: tosca.capabilities.Root
  valid_source_types: [ tosca.nodes.nfv.VDU ]
```

7.2.2 tosca.capabilities.nfv.HA

A node type that includes the HA capability indicates that it can be combine with other VDUs to provide High Availability capabilities using an nfv.relationships.HA relationship type.

Shorthand Name	HA
Type Qualified Name	tosca:HA
Type URI	tosca.capabilities.nfv.HA

7.2.2.1 Properties

Name	Required	Type	Constraints	Description
N/A	N/A	N/A	N/A	N/A

7.2.2.2 Definition

```
tosca.capabilities.nfv.HA:
  derived_from: tosca.capabilities.Root
  valid_source_types: [ tosca.nodes.nfv.VDU ]
```


7.2.3 tosca.capability.nfv.HA.ActiveActive

This capability type represents an ability to participate in an Active/Active redundancy model where two instances of the same VDU will co-exist with continuous data synchronization.

7.2.3.1 Definition

```
tosca.capabilities.nfv.HA.ActiveActive
  derived_from: tosca.capabilities.nfv.HA
```

7.2.4 tosca.capabilities.nfv.HA.ActivePassive

This capability type represents an ability to participate in an Active/Passive redundancy model where two instances of the same VDU will co-exists without any data synchronization.

7.2.4.1 Definition

```
tosca.capabilities.nfv.HA.ActivePassive:
  derived_from: tosca.capabilities.nfv.HA
```

7.2.5 tosca.capabilities.nfv.Metric

A node type that includes the Metric capability indicates that it can be monitored using an nfv.relationships.Monitor relationship type.

Shorthand Name	Metric
Type Qualified Name	tosca:Metric
Type URI	tosca.capabilities.nfv.Metric

7.2.5.1 Properties

Name	Required	Type	Constraints	Description
N/A	N/A	N/A	N/A	N/A

7.2.5.2 Definition

```
tosca.capabilities.nfv.Metric:
  derived_from: tosca.capabilities.Root
```

7.3 Relationship Types

7.3.1 tosca.relationships.nfv.VirtualBindsTo

This relationship type represents an association relationship between VDU and CP node types.

Shorthand Name	VirtualBindsTo
Type Qualified Name	tosca: VirtualBindsTo
Type URI	tosca.relationships.nfv. VirtualBindsTo

7.3.1.1 Definition

```
tosca.relationships.nfv. VirtualBindsTo:
  derived_from: tosca.relationships.ConnectsTo
  valid_target_types: [ tosca.capabilities.nfv.VirtualBindable]
```

7.3.2 tosca.relationships.nfv.HA

This relationship type represents a high-availability association between VDUs.

Shorthand Name	HA
Type Qualified Name	tosca:HA
Type URI	tosca.relationships.nfv.HA

7.3.2.1 Definition

```
tosca.relationships.nfv.HA:
  derived_from: tosca.relationships.Root
  valid_target_types: [ tosca.capabilities.nfv.HA]
```

7.3.3 tosca.relationships.nfv.Monitor

This relationship type represents an association relationship to the Metric capability of VDU node types.

Shorthand Name	Monitor
Type Qualified Name	tosca:Monitor
Type URI	tosca.relationships.nfv.Monitor

7.3.3.1 Definition

```
tosca.relationships.nfv.Monitor:
  derived_from: tosca.relationships.ConnectsTo
  valid_target_types: [ tosca.capabilities.nfv.Metric]
```

7.4 Node Types

7.4.1 tosca.nodes.nfv.VNF

The NFV VNF Node Type represents a Virtual Network Function as defined by [ETSI GS NFV-MAN 001 v1.1.1]. It is the default type that all other VNF Node Types derive from. This allows for all VNF nodes to have a consistent set of features for modeling and management (e.g., consistent definitions for requirements, capabilities and lifecycle interfaces).

```
nfv.nodes.VNF:
  derived_from: tosca.nodes.Root   # Or should this be its own top-level type?
  properties:
    id:
      type: string
      description: ID of this VNF
    vendor:
      type: string
      description: name of the vendor who generate this VNF
    version:
      type: version
      description: version of the software for this VNF
    requirements:
      - virtualLink:
         capability: tosca.capabilities.nfv.VirtualLinkable
```

7.4.2 tosca.nodes.nfv.VDU

The NFV vdu node type represents a logical vdu entity as defined by **[ETSI GS NFV-MAN 001 v1.1.1].**

Shorthand Name	VDU
Type Qualified Name	tosca:VDU
Type URI	tosca.nodes.nfv.VDU

7.4.2.1 Capabilities

Name	Type	Constraints	Description
monitoring_parameter	nvf.Metric	None	Monitoring parameter, which can be tracked for a VNFC based on this VDU Examples include: memory-consumption, CPU-utilisation, bandwidth-consumption, VNFC downtime, etc.
high_availability	nvf.HA		Defines ability to ensure high availability.
binding	tosca.Bindable		

7.4.2.2 Definition

```
tosca.nodes.nfv.VDU:
  derived_from: tosca.nodes.SoftwareComponent
```



```
  properties:

  capabilities:
    high_availability:
       type: nfv.capabilities.HA
    Virtualbinding:
      type: tosca.capabilities.nfv.VirtualBindable
     monitoring_parameter:
      type: nfv.capabilities.Metric
  requirements:
    - high_availability:
        capability: nfv.capabilities.HA
        relationship: nfv.relationships.HA
        occurrences: [ 0, 1 ]
    - host:
        capability: tosca.capabilities.Container
        node: tosca.nodes.Compute
        relationship: tosca.relationships.HostedOn
```

7.4.3 tosca.nodes.nfv.CP

The NFV CP node represents a logical connection point entity as defined by **[ETSI GS NFV-MAN 001 v1.1.1].** A connection point may be, for example, a virtual port, a virtual NIC address, a physical port, a physical NIC address or the endpoint of an IP VPN enabling network connectivity. It is assumed that each type of connection point will be modeled using subtypes of the CP type.

Shorthand Name	CP
Type Qualified Name	tosca:CP
Type URI	tosca.nodes.nfv.CP

7.4.4 Properties

Name	Required	Type	Constraints	Description
type	yes	string	None	This may be, for example, a virtual port, a virtual NIC address, a physical port, a physical NIC address or the endpoint of an IP VPN enabling network connectivity.

7.4.5 Attributes

Name	Required	Type	Constraints	Description
IP_address	no	string	None	The actual virtual NIC address that is been assigned when instantiating the connection point

7.4.6 Definition

```
tosca.nodes.nfv.CP:
  derived_from: tosca.nodes.Root
  properties:
    type:
  type:string
  required:false
  requirements:
    - virtualLink:
        capability: tosca.capabilities.VirtualLinkable
    - virtualbinding
        capability: tosca.capabilities.nfv.Virtualbindable
  attributes:
    IP_address:
     type: string
     required: false
```

7.4.7 Additional Requirement

8 TOSCA template for VLD

8.1 tosca.nodes.nfv.VL

The NFV VL node type represents a logical virtual link entity as defined by **[ETSI GS NFV-MAN 001 v1.1.1].** It is the default type from which all other virtual link types derive.

Shorthand Name	VL
Type Qualified Name	tosca:VL
Type URI	tosca.nodes.nfv.VL

8.1.1 Properties

Name	Required	Type	Constraints	Description
vendor	yes	string	None	Vendor generating this VLD

8.1.2 Attributes

8.1.3 Definition

```
tosca.nodes.nfv.VL:
  derived_from: tosca.nodes.Root
  properties:
    vendor:
      type:string
      required:true
       description: name of the vendor who generate this VL
  capabilities:
    virtual_linkable:
      type: nfv.capabilities.VirtualLinkable
```

8.1.4 Additional Requirement

8.2 tosca.nodes.nfv.VL.ELine

The NFV VL.ELine node represents an E-Line virtual link entity.

```
tosca.nodes.nfv.VL.ELine:
  derived_from: tosca.nodes.nfv.VL
```

```
  capabilities:
    virtual_linkable:
      occurrences: 2
```

8.3 tosca.nodes.nfv.VL.ELAN

The NFV VL.ELan node represents an E-LAN virtual link entity.

```
tosca.nodes.nfv.VL.ELAN:
  derived_from: tosca.nodes.nfv.VL
```

8.4 tosca.nodes.nfv.VL.ETree

The NFV VL.ETree node represents an E-Tree virtual link entity.

```
tosca.nodes.nfv.VL.ETree:
  derived_from: tosca.nodes.nfv.VL
```


9 TOSCA template for VNFFGD

10# Conformance

The last numbered section in the specification must be the Conformance section. Conformance Statements/Clauses go here. [Remove # marker]

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Chris Lauwers (lauwers@ubicity.com), Ubicity

Derek Palma (dpalma@vnomic.com), Vnomic

Matt Rutkowski (mrutkows@us.ibm.com), IBM

Shitao li (lishitao@huawei.com), Huawei

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD01, Rev01	2015-2-26	Shitao li, Huawei	● Adding clause 1, the introduction about this profile ● Adding clause 2, summary of key TOSCA concepts ● Adding clause 3, deployment template in NFV ● Adding clause 4, general mapping between TOSCA and NFV deployment template ● Adding clause 5, describes the main idea about using a service template for NFV NSD
WD01, Rev02	2015-4-15	Shitao li, Huawei	● Changing the NSD example used in clause 5 ● Changing the TOSCA model for NSD in figure 3 in clause 5, consider a VNF and its connection point as a subsystem of a NS ● Adding the TOSCA template example for NSD in clause 5.1 ● Adding NFV specific service properties for NSD in clause 5.2, the main properties are id ,vender and version ● Adding new capability tosca.capabilities.nfv.VirtualLinkable in clause 5.3 ● Adding new relationship type tosca.relationships.nfv.VirtualLinkTo in clause 5.4, which used between connection point and virtual link node types. ● Adding clause 6, TOSCA data model for VNFD ● Adding clause 6.1, node template substitution mapping for a VNF ● Adding NFV specific service properties for VNFD in clause 6.2, the main properties are id ,vender and version ● Adding new node type tosca.nodes.nfv.vdu in clause 6.3 ● Adding new node type tosca.nodes.nfv.CP in clause 6.4 ● Adding clause 7, TOSCA template for VLD (virtual link descriptor) ● Adding new node type tosca.nodes.nfv.VL in clause 7.1
WD01, Rev03	2015-5-5	Shitao li, Huawei Chris Lauwers	● Adding clause 3 for NFV overview ● Adding namespace for tosca-nfv- profile in clause 5.1 ● Deleting the NFV specific service properties for

			NSD and VNFD • Adding capability type definitions for VNF in clause 7.2(VirtualBindable, HA, HA.ActiveActive, HA.ActivePassive, Metric) • Adding relationship type definitions for VNF in clause 7.3(VirtualBindsTo, nfv.HA, nfv.Monitor) • Adding default VNF node type definition in clause 7.4.1 • Changing the VDU node type definition in clause 7.4.2(treat HA and monitor parameters as capabilities) • Adding new node types definition for VL.Eline, VL.ELAN and VL.ETree in clause 8.2, 8.3 and 8.4.
WD01, Rev04	2015-5-13	Chris Lauwers	• Formatting changes

