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

Committee Specification Draft 04

11 May 2017

Specification URIs

This version:

http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.pdf (Authoritative)
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.doc

Previous version:

http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.pdf (Authoritative)
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.html
http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.doc

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

John Crandall (jcrandal@brocade.com), Brocade

Editors:

Shitao Li (lishitao@huawei.com), Huawei Technologies Co., Ltd.
John Crandall (jcrandal@brocade.com), Brocade

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.

Declared XML namespaces:

- http://docs.oasis-open.org/tosca/ns/simple/yaml/1.0/nfv/1.0/

Abstract:

The TOSCA NFV profile specifies a Network Functions Virtualisation (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/.

This Committee Specification Draft is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. 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).

Note that any machine-readable content (aka Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

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 and John Crandall. 11 May 2017. OASIS Committee Specification Draft 04. http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html. Latest version: http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

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 IPR Policy

This Committee Specification Draft is being developed under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established.

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).

1.2 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]**.

1.3 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.

[**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

[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 Template Specification

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 Architecture & Concept 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 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 and virtual network function using NSDs and VNFDs, respectfully.

3.1 Deployment Template in NFV

The deployment template in NFV fully describes the attributes and requirements necessary to realize such a Network Service. NFV Orchestrator (NFVO) manages the lifecycle of network service, manage the VNF lifecycle via the interface exposed by the VNF Manager (VNFM), and manages virtualized resources via the interfaces exposed by the VIM.

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.

3.2 Network Services Descriptor

Editor note: A section describing ETSI NFV architecture & concept of NSD (IFA014). And, subsection describing some of the basic terminologies.

The Network Service Descriptor (NSD) is a deployment template which consists of information used by the NFV Orchestrator (NFVO) for life cycle management of an NS [ETSI GS NFV-IFA 014]. The description of a NS as used by the NFV Management and Orchestration (MANO) functions to deploy an NS instance includes or references 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".

3.2.1 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. In this example, the network service includes three VNFs with two connections points (CP1 and CP14). Each VNF exposes different number of connection points and connect through a virtual links, such as VL1={CP1, CP4}; VL2={CP5, CP8,CP10}; VL3={CP6,CP7}; VL4={CP11, CP14}. Virtual Link connects two or more connection points. VNF Forwarding Graph represents the connections of the VNFs are connected through connection points and Virtual Links. Network Forwarding Path represents the flow where the packet will follow.

In this example, there are two VNF Forwarding Graph (VNFFG1 and VNFFG2), where each of VNFFG has different Network Forwarding Path (NFP1 and NFP2).

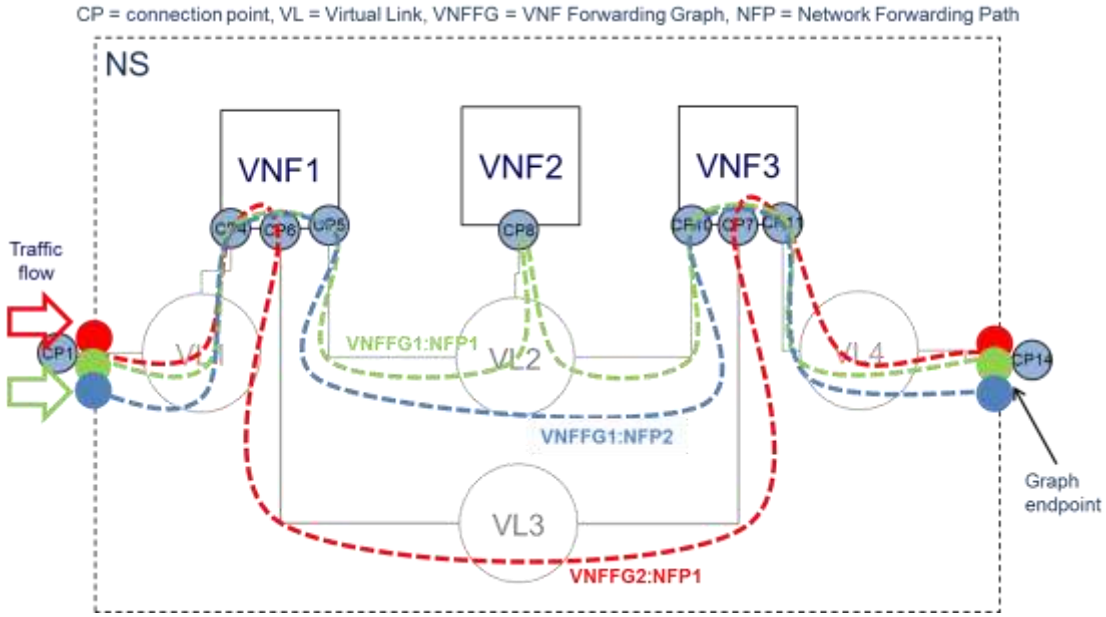


Figure 3.2.1-1: Example network connectivity topology graph

•

3.3 VNFD: Virtualized Network Function Descriptor

A VNFD is a deployment template which describes a VNF in terms of deployment and operational behavior requirements. It also contains connectivity, interface and virtualized resource requirements [ETSI GS NFV-IFA 011]. The main parts of the VNFD are the following:

- VNF topology: it is modeled in a cloud agnostic way using virtualized containers and their connectivity. Virtual Deployment Units (VDU) describe the capabilities of the virtualized containers, such as virtual CPU, RAM, disks; their connectivity is modeled with VDU Connection Point Descriptors (VduCpd), Virtual Link Descriptors (Vld) and VNF External Connection Point Descriptors (VnfExternalCpd);
- VNF deployment aspects: they are described in one or more deployment flavours, including configurable parameters, instantiation levels, placement constraints (affinity / antiaffinity), minimum and maximum VDU instance numbers. Horizontal scaling is modeled with scaling aspects and the respective scaling levels in the deployment flavours;
- VNF lifecycle management (LCM) operations: describes the LCM operations supported per deployment flavour, and their input parameters; Note, that the actual LCM implementation resides in a different layer, namely referring to additional template artifacts.

Editor Note: VNF LCM operation modeling in TOSCA is still under discussion.

4 TOSCA Modeling Principles & Data Model

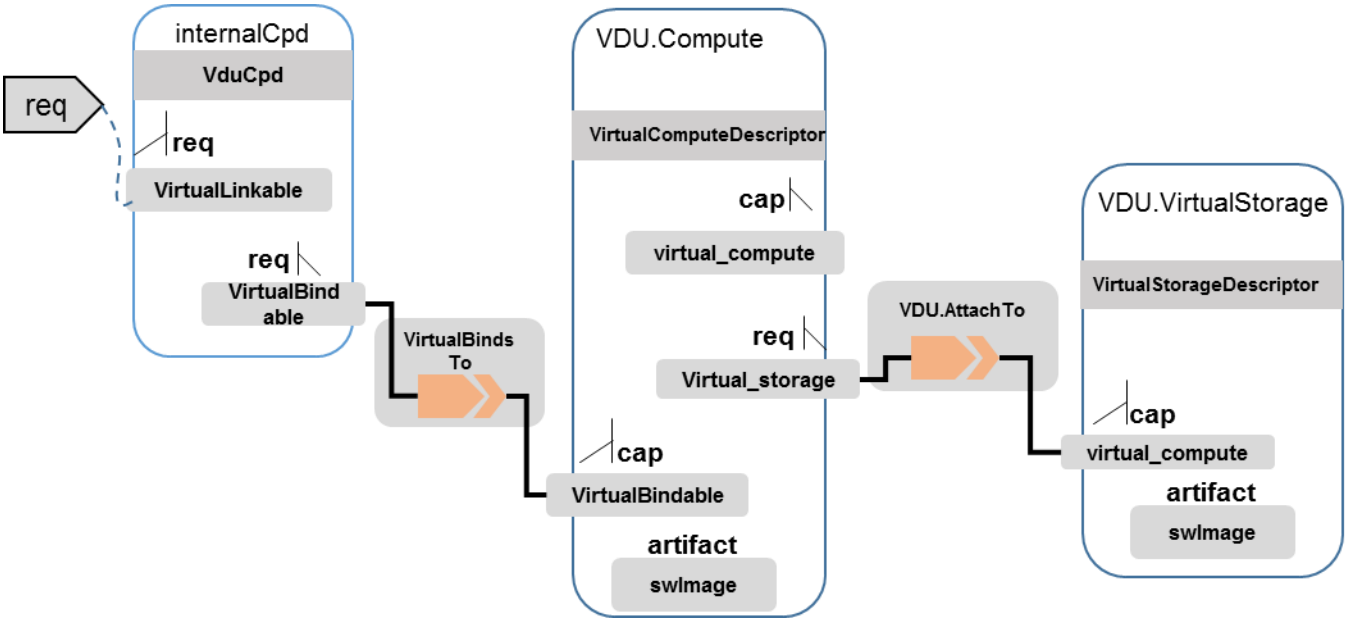
Editor Note: This section describing TOSCA modeling principles and data model for NFV, where the type, properties, capabilities, requirements, and relationships, etc. may/should/shall be used based on [TOSCA-1.0] and [TOSCA-Simple-Profile-YAML V1.0], or new type based on ETSI NFV requirements, etc.

4.1 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 specification.

Alias	Target Namespace	Specification Description
tosca_simple_profile_for_nfv_1_0	http://docs.oasis-open.org/tosca/ns/simple/yaml/1.0/nfv/1.0/	The TOSCA Simple Profile for NFV v1.0 target namespace and namespace alias.

4.2 VDU.Compute



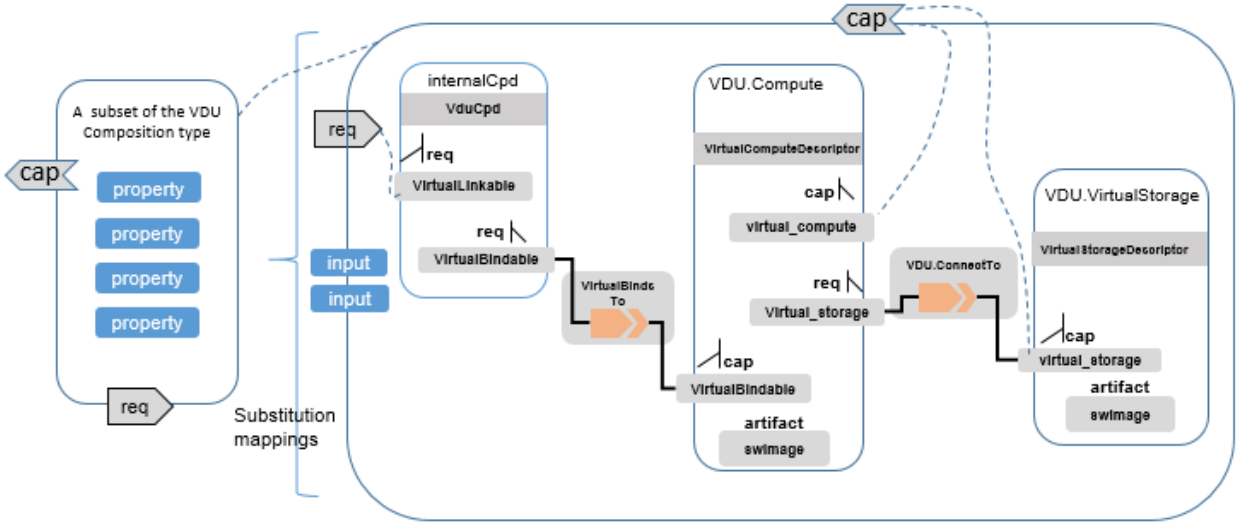
EDITOR NOTE: FFS. Document use of decoration of the VDU.Compute node with additional capabilities type (which carry properties is the prefer method) e.g. additional processor architecture requirements to existing VDU.Compute

4.3 VDU design by using TOSCA composition

Node template substitution for model composition feature as specified in TOSCA-Simple-Profile-YAML is used to abstract a subsystem as a component of another application. The details for such subsystem can

be defined in a separate template file that can be used for substituting the more abstract representations in the application level template file.

For a VNF descriptor design, VDU can be considered as the subsystem of a VNF descriptor, a standalone TOSCA service template can be used to define the tolopogy of the VDU, and then the TOSCA compostion can be used to abstract the VDU as a new node type.



Editor's note: `the definition of tosca.nodes.nfv.VDUComposition is FFS.`

5 VNF Descriptor Template for NFV

5.1 Introduction

The VNF Descriptor (VNFD) describes the topology of the VNF by means of ETSI NFV IFA011 [IFA011] terms such as VDUs, Connection Points, Virtual Links, External Connection Points, Scaling Aspects, Instantiation Levels and Deployment Flavours.

The VNFD (VNF Descriptor) is read by both the NFVO and the VNFM. It represents the contract & interface of a VNF and ensures the interoperability across the NFV functional blocks.

The main parts of the VNFD are the following:

- VNF topology: it is modeled in a cloud agnostic way using virtualized containers and their connectivity. Virtual Deployment Units (VDU) describe the capabilities of the virtualized containers, such as virtual CPU, RAM, disks; their connectivity is modeled with VDU Connection Point Descriptors (VduCpd), Virtual Link Descriptors (Vld) and VNF External Connection Point Descriptors (VnfExternalCpd);
- VNF deployment aspects: they are described in one or more deployment flavours, including instantiation levels, supported LCM operations, VNF LCM operation configuration parameters, placement constraints (affinity / antiaffinity), minimum and maximum VDU instance numbers, and scaling aspect for horizontal scaling.

5.2 TOSCA model for VNFD

The following table defines the TOSCA Type “derived from” values that SHALL be used when using the TOSCA Simple Profile for NFV version 1.0 specification [TOSCA-Simple-Profile-NFV-v1.0] for NFV VNFD.

ETSI NFV Element [IFA011]	**TOSCA VNFD [TOSCA-Simple-Profile-NFV-v1.0]**	**Derived from**
VNF	tosca.nodes.nfv.VNF	tosca.nodes.Root
VDU	tosca.nodes.nfv.VDU	tosca.nodes.Root
Cpd (Connection Point)	tosca.nodes.nfv.Cpd	tosca.nodes.Root
VduCpd (internal connection point)	tosca.nodes.nfv.VduCpd	tosca.nodes.nfv.Cpd
VnfVirtualLinkDesc (Virtual Link)	tosca.nodes.nfv.VnfVirtualLinkDesc	tosca.nodes.Root
VnfExtCpd (External Connection Point)	tosca.nodes.nfv.VnfExtCpd	tosca.nodes.Root
Virtual Storage		
Virtual Compute	tosca.capabilities.nfv.VirtualCompute	tosca.capabilities.Root
Software Image	tosca.artifacts.nfv.SwImage	tosca.artifacts.Deploy

		ment.Image
Deployment Flavour		
Scaling Aspect		
Element Group		
Instantiation Level		

5.2.1 Additional Requirements

This Profile's Node Type definitions are utilizing existing TOSCA grammar to:

- change the status of an inherited property (i.e., a property's "status" keyname's value)
- change the occurrences of inherited requirements or capabilities (e.g., by turning off a requirement by setting the occurrences keyname's value to [0,0]

However, these features are not explicitly supported in TOSCA Simple Profile in YAML version 1.1, but the NFV work group has raised this as a requirement for the version 1.2 Simple Profile in YAML and has been assured this will appear as a supported feature of the grammar in that version.

5.3 Data Types

5.3.1 tosca.datatype.nfv.L2AddressData

Editor Note: Further discussion with ETSI IFA/SOL WG to defines these values.

Shorthand Name	L2AddressData
Type Qualified Name	tosca:tosca.datatype.nfv.L2AddressData
Type URI	tosca.datatype.nfv.L2AddressData

5.3.1.1 Properties

TBD

Name	Required	Type	Constraints	Description

5.3.1.2 Definition

```
TBD
```

5.3.1.3 Examples

TBD

5.3.2 tosca.datatypes.nfv.L3AddressData

The L3AddressData type is a complex TOSCA data type used to describe L3AddressData information element as defined in [ETSI GS NFV-IFA 011], it provides the information on the IP addresses to be assigned to the connection point instantiated from the parent Connection Point Descriptor.

Shorthand Name	L3AddressData
Type Qualified Name	tosca: L3AddressData
Type URI	tosca.datatypes.nfv.L3AddressData

5.3.2.1 Properties

Name	Required	Type	Constraints	Description
ip_address_assignment	yes	Boolean		Specify if the address assignment is the responsibility of management and orchestration function or not. If it is set to True, it is the management and orchestration function responsibility.
floating_ip_activated	yes	Boolean		Specify if the floating IP scheme is activated on the Connection Point or not.
ip_address_type	no	string	Valid values: ipv4 , ipv6	Define address type. The address type should be aligned with the address type supported by the layer_protocol properties of the parent VnfExtCpd
number_of_ip_address	no	Integer		Minimum number of IP addresses to be assigned.

5.3.2.2 Definition

The TOSCA L3AddressData data type is defined as follows:

```
tosca.datatypes.nfv.L3AddressData:
  derived_from: tosca.datatypes.Root
  properties:
    ip_address_assignment:
      type: Boolean
      required: true
    floating_ip_activated:
      type: Boolean
      required: true
    ip_address_type:
```

```
      type: string
      required: false
      constraints:
        - valid_values: [ipv4, ipv6]
    number_of_ip_address:
      type: integer
      required: false
```

5.3.2.3 Examples

Example usage of the L3AddressData data type:

```
<some_tosca_entity>:
   properties:
      l3_address_data:
          ip_address_assignment: true
          floating_ip_activated: true
          ip_address_type: ipv4
          number_of_ip_address: 4
```

5.3.3 tosca.datatypes.nfv.AddressData

The AddressData type is a complex TOSCA data type used to describe AddressData information element as defined in [ETSI GS NFV-IFA 011], it provides information on the addresses to be assigned to the connection point(s) instantiated from a Connection Point Descriptor.

Shorthand Name	AddressData
Type Qualified Name	tosca: AddressData
Type URI	tosca.datatypes.nfv.AddressData

5.3.3.1 Properties

Name	**Required**	**Type**	**Constraints**	**Description**
address_type	yes	string	Valid values: mac_address ip_address	Describes the type of the address to be assigned to the connection point instantiated from the parent Connection Point Descriptor. The content type shall be aligned with the address type supported by the layerProtocol property of the parent Connection Point Descriptor.
l2_address_data	no	tosca.datatypes.nfv.L2AddressData	Shall be present when the addressType is mac_address.	Provides the information on the MAC addresses to be assigned to the connection point(s) instantiated from the parent Connection Point Descriptor.

Name	Required	Type	Constraints	Description
l3_address_data	no	L3AddressData	Shall be present when the addressType is ip_address.	Provides the information on the IP addresses to be assigned to the connection point instantiated from the parent Connection Point Descriptor.

5.3.3.2 Definition

The TOSCA AddressData data type is defined as follows:

```
tosca.datatypes.nfv.AddressData:
  derived_from: tosca.datatypes.Root
  properties:
    address_type:
      type: string
      required: true
      constraints:
        - valid_values: [mac_address, ip_address]
    l2_address_data:
      type: tosca.datatypes.nfv.L2AddressData # empty in "GS NFV IFA011 V0.7.3"
      required: false
    l3_address_data:
      type: tosca.datatypes.nfv.L3AddressData
      required: false
```

5.3.3.3 Examples

Example usage of the AddressData data type:

```
<some_tosca_entity>:
   properties:
     address_Data:
        address_type: IP address
        l3_address_data:
            ip_address_assignment: true
            floating_ip_activated: true
            ip_address_type: IPv4 address
            number_of_ip_Address: 4
```

5.3.4 tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements

The VirtualNetworkInterfaceRequirements type is a complex TOSCA data type used to describe VirtualNetworkInterfaceRequirements information element as defined in [ETSI GS NFV-IFA 011], it provides the information to specify requirements on a virtual network interface realising the CPs instantiated from this CPD.

Shorthand Name	VirtualNetworkInterfaceRequirements
Type Qualified Name	tosca: VirtualNetworkInterfaceRequirements
Type URI	tosca.datatypes.nfv. VirtualNetworkInterfaceRequirements

5.3.4.1 Properties

Name	Required	Type	Constraints	Description
name	no	string		Provides a human readable name for the requirement.
description	no	string		Provides a human readable description of the requirement.
support_mandatory	yes	boolean	none	Indicates whether fulfilling the constraint is mandatory (TRUE) for successful operation or desirable (FALSE).
requirement	yes	Not specified		Specifies a requirement such as the support of SR-IOV, a particular data plane acceleration library, an API to be exposed by a NIC, etc.

5.3.4.2 Definition

The TOSCA VirtualNetworkInterfaceRequirements data type is defined as follows:

```
tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements:
  derived_from: tosca.datatypes.Root
  properties:
    name:
      type: string
      required: false
    description:
      type: string
      required: false
    support_mandatory:
      type: boolean
      required: true
    requirement:
      type:           # not specified
      required: true
```

5.3.4.3 Examples

Example usage of the VirtualNetworkInterfaceRequirements data type:

```
<some_tosca_entity>:
   properties:
      virtual_network_interface_requirements:
          name: SR-IOV
          description: support of SR-IOV
          support_mandatory: true
```

5.3.5 tosca.datatypes.nfv.ConnectivityType

The TOSCA ConnectivityType type is a complex TOSCA data type used to describe ConnectivityType information element as defined in [ETSI GS NFV-IFA 011].

Shorthand Name	ConnectivityType
Type Qualified Name	tosca: ConnectivityType
Type URI	tosca.datatypes.nfv. ConnectivityType

5.3.5.1 Properties

Name	Required	Type	Constraints	Description
layer_protocol	yes	string	Valid values: ethernet, mpls, odu2, ipv4, ipv6, pseudo_wire	Identifies the protocol this VL gives access to (ethernet, mpls, odu2, ipv4, ipv6, pseudo_wire).
flow_pattern	no	string		Identifies the flow pattern of the connectivity (Line, Tree, Mesh).

5.3.5.2 Definition

The TOSCA ConnectivityType data type is defined as follows:

```
tosca.datatypes.nfv. ConnectivityType:
  derived_from: tosca.datatypes.Root
  properties:
    layer_protocol:
      type: string
      required: yes
      constraints:
        - valid_values: [ethernet, mpls, odu2, ipv4, ipv6, pseudo_wire ]
    flow_pattern:
```

```
      type: string
      required: false
```

5.3.5.3 Examples

Example usage of the VirtualNetworkInterfaceRequirements data type:

```
<some_tosca_entity>:
   properties:
      Connectivity_Type:
          layer_protocal: lpv4
          flow_pattern: Line
```

5.3.6 tosca.datatypes.nfv.RequestedAdditionalCapability

RequestAdditionalCapability describes additional capability for a particular VDU.

Shorthand Name	RequestedAdditionalCapability
Type Qualified Name	tosca: RequestedAdditionalCapability
Type URI	tosca.datatypes.nfv.RequestedAdditionalCapability
derived_from	tosca.datatype.Root

5.3.6.1 Properties

Name	Required	Type	Constraints	Description
request_additional_capability_name	yes	string		Identifies a requested additional capability for the VDU.
support_mandatory	yes	boolean		Indicates whether the requested additional capability is mandatory for successful operation.
min_requested_additional_capability_version	no	string		Identifies the minimum version of the requested additional capability.
preferred_requested_additinal_capability_version	no	string		Identifies the preferred version of the requested additional capability.
target_performance_parameters	yes	map of string		Identifies specific attributes, dependent on the requested additional capability type.

5.3.6.2 Definition

```
tosca.datatypes.nfv.RequestedAdditionalCapability:
  derived_from: tosca.datatypes.Root
  properties:
    #name:
      # key of containing map
    support_mandatory:
      type: boolean
      required: true
    min_requested_additional_capability_version:
      type: string
      required: false
    preferred_requested_additional_capability_version:
      type: string
      required: false
    requested_additional_capability_name:
      type: string
      required: true
    target_performance_parameters:
      type: map
      entry_schema:
        type: string
      required: true
```

5.3.6.3 Examples

TBD

5.3.6.4 Additional Requirements

None

5.3.7 tosca.datatypes.nfv.VirtualMemory

VirtualMemory describes virtual memory for a particular VDU.

Shorthand Name	VirtualMemory
Type Qualified Name	tosca:VirtualMemory
Type URI	tosca.datatypes.nfv.VirtualMemory
derived_from	tosca.datatype.Root

5.3.7.1 Properties

Name	Required	Type	Constraints	Description
virtual_mem_size	yes	scalar-unit.size	number	Amount of virtual memory.
virtual_mem_oversubscription_polity	no	string		The memory core oversubscription policy in terms of virtual memory to physical memory on the platform. The cardinality can be 0 during the allocation request, if no particular value is requested.
numa_enabled	no	boolean		It specifies the memory allocation to be cognisant of the relevant process/core allocation. The cardinality can be 0 during the allocation request, if no particular value is requested.

5.3.7.2 Definition

```
tosca.datatypes.nfv.VirtualMemory:
    derived_from: tosca.datatypes.Root
    properties:
      virtual_mem_size:
        type: scalar-unit.size # Number
        required: true
      virtual_mem_oversubscription_policy:
        type: string
        required: false
      numa_enabled:
        type: boolean
        required: false
```

5.3.7.3 Examples

TBD

5.3.7.4 Additional Requirements

None

5.3.8 tosca.datatypes.nfv.VirtualCpu

VirtualMemory describes virtual memory for a particular VDU.

Shorthand Name	VirtualCpu
Type Qualified Name	tosca:VirtualCpu
Type URI	tosca.datatypes.nfv.VirtualCpu
derived_from	tosca.datatype.Root

5.3.8.1 Properties

Name	Required	Type	Constraints	Description
cpu_architecture	no	string		CPU architecture type. Examples are x86, ARM.
num_virtual_cpu	yes	integer		Number of virtual CPU's
virtual_cpu_clock	no	scalar-unit.frequency		Minimum virtual CPU clock rate
virtual_cpu_oversubscription_policy	no	string		CPU core oversubscription policy
virtual_cpu_pinning	no	tosca.datatypes.nfv.VirtualCpuPinning		The virtual CPU pinning configuration for the virtualized compute resource.

5.3.8.2 Definition

```
tosca.datatypes.nfv.VirtualCpu:
  derived_from: tosca.datatypes.Root
  properties:
    cpu_architecture:
      type: string
      required: false
    num_virtual_cpu:
      type: integer
      required: true
    virtual_cpu_clock:
      type: scalar-unit.frequency
      required: false
    virtual_cpu_oversubscription_policy:
      type: string
      required: false
    virtual_cpu_pinning:
```

```
      type: tosca.datatypes.nfv.VirtualCpuPinning
      required: false
```

5.3.8.3 Examples

TBD

5.3.6.1 Additional Requirements

None

5.3.9 tosca.datatypes.nfv.VirtualCpuPinning

VirtualCpuPinning describes CPU pinning configuration for a particular CPU.

Shorthand Name	VirtualCpuPinning
Type Qualified Name	tosca:VirtualCpuPinning
Type URI	tosca.datatypes.nfv.VirtualCpuPinning
derived_from	tosca.datatype.Root

5.3.9.1 Properties

Name	Required	Type	Constraints	Description
cpu_pinning_policy	no	string	Static or dynamic	Indicates the policy for CPU pinning.
cpu_pinning_map	no	map		If cpuPinningPolicy is defined as "static", the cpuPinningMap provides the map of pinning virtual CPU cores to physical CPU cores/threads

5.3.9.2 Definition

```
tosca.datatypes.nfv.VirtualCpuPinning:
  derived_from: tosca.datatypes.Root
  properties:
    cpu_pinning_policy:
      type: string # CpuPinningPolicy
      constraints:
        - valid_values: [ static, dynamic ]
      required: false
    cpu_pinning_map:
      type: map
```

```
      entry_schema:
        type: string
      required: false
```

5.3.9.3 Examples

TBD

5.3.9.4 Additional Requirements

None

5.3.10 tosca.datatypes.nfv.VnfcConfigurableProperties

VnfcconfigurableProperties describes VirtualCpuPinning describes additional configurable properties of a VNFC

Shorthand Name	VnfcconfigurableProperties
Type Qualified Name	tosca: VnfcconfigurableProperties
Type URI	tosca.datatypes.nfv.VnfcconfigurableProperties
derived_from	tosca.datatype.Root

5.3.10.1 Properties

Name	**Required**	**Type**	**Constraints**	**Description**
additional_vnfc_configurable_properties	no	map		Described additional configuration for VNFC

5.3.10.2 Definition

```
tosca.datatypes.nfv.VnfcConfigurableProperties:
  derived_from: tosca.datatypes.Root
  properties:
    additional_vnfc_configurable_properties:
      type: map
      entry_schema:
        type: string
      required: false
```

5.3.10.3 Examples

TBD

5.3.10.4 Additional Requirements

None

5.4 Artifact types

5.4.1 tosca.artifacts.nfv.SwImage

Shorthand Name	SwImage
Type Qualified Name	tosca:SwImage
Type URI	tosca.artifacts.nfv.SwImage
derived_from	tosca.artifacts.Deployment.Image

5.4.1.1 Properties

Name	Required	Type	Constraints	Description
name	yes	string		Name of this software image
version	yes	string		Version of this software image
checksum	yes	string		Checksum of the software image file
container_format	yes	string		The container format describes the container file format in which software image is provided.
disk_format	yes	string		The disk format of a software image is the format of the underlying disk image
min_disk	yes	scalar-unit.size		The minimal disk size requirement for this software image.
min_ram	no	scalar-unit.size		The minimal RAM requirement for this software image.
Size	yes	scalar-unit.size		The size of this software image
sw_image	yes	string		A reference to the actual software image within VNF Package, or url.
operating_system	no	string		Identifies the operating system used in the software image.
supported _virtualization_enviroment	no	list		Identifies the virtualization environments (e.g. hypervisor) compatible with this software image

5.4.1.2 Definition

```
tosca.artifacts.nfv.SwImage:
  derived_from: tosca.artifacts.Deployment.Image
  properties or metadata:
    #id:
      # node name
    name:
      type: string
      required: true
    version:
      type: string
      required: true
    checksum:
      type: string
      required: true
    container_format:
      type: string
      required: true
    disk_format:
      type: string
      required: true
    min_disk:
      type: scalar-unit.size # Number
      required: true
    min_ram:
      type: scalar-unit.size # Number
      required: false
    size:
      type: scalar-unit.size # Number
      required: true
    sw_image:
      type: string
      required: true
    operating_system:
      type: string
      required: false
    supported_virtualisation_environments:
      type: list
      entry_schema:
        type: string
      required: false
```

5.5 Capabilities Types

5.5.1 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

5.5.1.1 Properties

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

5.5.1.2 Definition

```
tosca.capabilities.nfv.VirtualBindable:
  derived_from: tosca.capabilities.Node
```

5.5.2 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

5.5.2.1 Properties

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

5.5.2.2 Definition

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

5.5.3 tosca.capabilites.nfv.VirtualCompute

Shorthand Name	VirtualCompute
Type Qualified Name	tosca: VirtualCompute
Type URI	tosca.capabilities.nfv.VirtualCompute
derived from	tosca.nodes.Root

5.5.3.1 Properties

Name	Required	Type	Constraints	Description
request_additional_capabilities	No	tosca.datatypes.nfv.RequestedAdditionalCapability		Describes additional capability for a particular VDU.
virtual_memory	yes	tosca.datatypes.nfv.VirtualMemory		Describes virtual memory of the virtualized compute
virtual_cpu	yes	tosca.datatypes.nfv.VirtualCpu		Describes virtual CPU(s) of the virtualized compute.

5.5.3.2 Definition

```
tosca.capabilities.nfv.VirtualCompute:
  derived_from: tosca.capabilities.Root
  properties:
    requested_additional_capabilities:
      type: map
      entry_schema:
         type: tosca.datatypes.nfv.RequestedAdditionalCapability
      required: false
    virtual_memory:
      type: tosca.datatypes.nfv.VirtualMemory
      required: true
    virtual_cpu:
      type: tosca.datatypes.nfv.VirtualCpu
      required: true
```

5.6 Requirements Types

5.7 Relationship Types

5.7.1 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

5.7.1.1 Definition

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

5.7.2 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

5.7.2.1 Definition

```
```

5.8 Interface Types

5.9 Node Types

5.9.1 tosca.nodes.nfv.vnfd

5.9.2 tosca.nodes.nfv.VDU.Compute

The TOSCA nfv.VDU.Compute node type represents the virtual compute part of a VDU entity which it mainly describes the deployment and operational behavior of a VNF component (VNFC), as defined by **[ETSI NFV IFA011].**

.

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

5.9.2.1 Properties

Name	Required	Type	Constraints	Description
name	yes	string		Human readable name of the Vdu
description	yes	string		Human readable description of the Vdu
boot_order	no	list of string		The key indicates the boot index (lowest index defines highest boot priority). The Value references a descriptor from which a valid boot device is created e.g. VirtualStorageDescriptor from which a VirtualStorage instance is created. If no boot order is defined the default boot order defined in the VIM or NFVI shall be used.
nfvi_constraints	no	list of string		Describes constraints on the NFVI for the VNFC instance(s) created from this Vdu. For example, aspects of a secure hosting environment for the VNFC instance that involve additional entities or processes. More software images can be attached to the virtualization container using virtual_storage.
configurable_properties	yes	map of tosca.datatypes.nfv.VnfcConfigurableProperties		Describes the configurable properties of all VNFC instances based on this VDU.

5.9.2.2 Attributes

None

5.9.2.3 Requirements

Name	Required	Type	Constraints	Description
virtual_storage	no	tosca.nodes.nfv.VDU.VirtualStorage		Describes storage requirements for a virtual_storage instance attached to the virtualization container created from virtual_compute defined for this vdu

5.9.2.4 Capabilities

Name	Type	Constraints	Description
virtual_compute	tosca.capabilities.nfv.VirtualCompute		Describes virtual compute resources capabilities.
monitoring_parameter	tosca.capabilities.nfv.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.
Virtual_binding	tosca.capabilities.nfv.VirtualBindable editor note: need to create a capability type		Defines ability of VirtualBindable

5.9.2.5 Definition

```
tosca.nodes.nfv.VDU.Compute:
  derived_from: tosca.nodes.Compute
  properties:
    name:
      type: string
      required: true
    description:
      type: string
      required: true
    boot_order:
      type: list # explicit index (boot index) not necessary, contrary to IFA011
      entry_schema:
        type: string
      required: false
    nfvi_constraints:
      type: list
      entry_schema:
        type: string
      required: false
    configurable_properties:
       type: map
       entry_schema:
          type: tosca.datatypes.nfv.VnfcConfigurableProperties
       required: true
```

```
attributes:
    private_address:
      status: deprecated
    public_address:
      status: deprecated
    networks:
      status: deprecated
    ports:
      status: deprecated
  capabilities:
    virtual_compute:
      type: tosca.capabilities.nfv.VirtualCompute
    virtual_binding:
      type: tosca.capabilities.nfv.VirtualBindable
    #monitoring_parameter:
      # modeled as ad hoc (named) capabilities in VDU node template
      # for example:
      #capabilities:
      #  cpu_load: tosca.capabilities.nfv.Metric
      # memory_usage: tosca.capabilities.nfv.Metric
    host: #Editor note: FFS. How this capabilities should be used in NFV Profile
      type: tosca.capabilities.Container
      valid_source_types: [tosca.nodes.SoftwareComponent]
        occurrences: [0,UNBOUNDED]
    endpoint:
        occurrences: [0,0]
    os:
        occurrences: [0,0]
    scalable: #Editor note: FFS. How this capabilities should be used in NFV
Profile
      type: tosca.capabilities.Scalable
    binding:
        occurrences: [0,UNBOUND]
  requirements:
    - virtual_storage:
        capability: tosca.capabilities.nfv.VirtualStorage
        relationship: tosca.relationships.nfv.VDU.AttachedTo
        node: tosca.nodes.nfv.VDU.VirtualStorage
        occurences: [ 0, UNBOUNDED ]
    - local_storage: #For NFV Profile, this requirement is deprecated.
        occurrences: [0,0]
```

```
artifacts:
  - sw_image:
      file:
      type: tosca.artifacts.nfv.SwImage
```

5.9.2.6 Artifact

Name	Required	Type	Constraints	Description
SwImage	Yes	tosca.artifacts.nfv.SwImage		Describes the software image which is directly loaded on the virtualization container realizing this virtual storage.

5.9.3 tosca.nodes.nfv.VDU.VirtualStorage

The NFV VirtualStorage node type represents a virtual storage entity which it describes the deployment and operational behavior of a virtual storage resources, as defined by **[ETSI NFV IFA011].**

[editor note] open issue: should NFV profile use the current storage model as described in YAML 1.1. Pending on Shitao proposal (see NFVIFA(17)000110 discussion paper)

[editor note] new relationship type as suggested in Matt presentation. Slide 8. With specific rules of "valid_target_type"

Shorthand Name	VirtualStorage
Type Qualified Name	tosca: VirtualStorage
Type URI	tosca.nodes.nfv.VDU.VirtualStorage
derived_from	tosca.nodes.Root

5.9.3.1 Properties

Name	Required	Type	Constraints	Description
id	yes			Unique identifier of the virtualStorage
type_of_storage	yes	string	volune, object	Type of virtualized storage resource
size_of_storage	yes	scalar-unit.size	number	Size of virtualized storage resource (in GB)
rdma_enabled	no	boolean		Indicate if the storage support RDMA

5.9.3.2 Attributes

None

5.9.3.3 Requirements

None

5.9.3.4 Capabilities

Name	Type	Constraints	Description
virtual_storage	tosca.capabilities.nfv.VirtualStorage Editor Note: Need to create tosca.capabilities.nfv.VirtualStorage capability type.		Defines the capabilities of virtual_storage.

5.9.3.5 Definition

```
tosca.nodes.nfv.VDU.VirtualStorage:
  derived_from: tosca.nodes.Root
  properties:
    #id:
      # node name
    type_of_storage:
      type: string
      required: true
    size_of_storage:
      type: scalar-unit.size
      required: true
    rdma_enabled:
      type: boolean
      required: false
  capabilities:
    virtual_storage:
      type: tosca.capabilities.nfv.VirtualStorage
  artifacts:
    - sw_image:
        file:
        type: tosca.artifacts.Deployment.Image
```

5.9.3.6 Artifact

Name	Required	Type	Constraints	Description
sw_image	yes	tosca.artifacts.Deployment.Image		Describes the software image which is directly loaded on the virtualization container realizing this virtual storage.

5.9.4 tosca.nodes.nfv.Cpd

The TOSCA nfv.Cpd node represents network connectivity to a compute resource or a VL as defined by [ETSI GS NFV-IFA 011]. This is an abstract type used as parent for the various Cpd types.

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

5.9.4.1 Properties

Name	Required	Type	Constraints	Description
layer_protocol	yes	string	Valid values: Ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire	Identifies which protocol the connection point uses for connectivity purposes
role	no	string	Editor's note: valid values: [root, leaf]	Identifies the role of the port in the context of the traffic flow patterns in the VNF or parent NS. For example a VNF with a tree flow pattern within the VNF will have legal cpRoles of ROOT and LEAF
description	no	string		Provides human-readable information on the purpose of the connection point (e.g. connection point for control plane traffic).
address_data	no	AddressData []		Provides information on the addresses to be assigned to the connection point(s) instantiated from this Connection Point Descriptor.

5.9.4.2 Attributes

Name	Required	Type	Constraints	Description

5.9.4.3 Requirements

None

5.9.4.4 Capabilities

None

5.9.4.5 Definition

```
tosca.nodes.nfv.Cpd:
  derived_from: tosca.nodes.Root
  properties:
    layer_protocol:
      type:string
      constraints:
        - valid_values: [ethernet, mpls, odu2, ipv4, ipv6, pseudo_wire ]
      required:true
    role: #Name in ETSI NFV IFA011 v0.7.3 cpRole
      type:string
      constraints:
        - valid_values: [ root, leaf ]
      required:flase
    description:
      type: string
      required: false
    address_data:
      type: list
      entry_schema:
        type: tosca.datatype.nfv.AddressData
      required:false
```

5.9.4.6 Additional Requirement

None.

5.9.5 tosca.nodes.nfv.VduCpd

The TOSCA nfv.VduCpd node type represents a type of TOSCA Cpd node and describes network connectivity between a VNFC instance (based on this VDU) and an internal VL as defined by [ETSI GS NFV-IFA 011].

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

5.9.5.1 Properties

Name	Required	Type	Constraints	Description
bitrate_requirement	no	integer		Bitrate requirement on this connection point.
virtual_network_interface_requirements	no	VirtualNetworkInterfaceRequirements []		Specifies requirements on a virtual network interface realising the CPs instantiated from this CPD.

5.9.5.2 Attributes

None

5.9.5.3 Requirements

Name	Required	Type	Constraints	Description
virtual_binding	yes	tosca.capabilities.nfv.VirtualBindable		Describe the requirement for binding with VDU
virtual_link	no	tosca.capabilities.nfv.VirtualLinkable		Describes the requirements for linking to virtual link

5.9.5.4 Definition

```
tosca.nodes.nfv.VduCpd:
  derived_from: tosca.nodes.nfv.Cpd
  properties:
     bitrate_requirement:
       type: integer
       required:false
     virtual_network_interface_requirements
       type: list
       entry_schema:
         type: VirtualNetworkInterfaceRequirements
       required:false
   requirements:
     - virtual_link:
```

```
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        node: tosca.nodes.nfv.VnfVirtualLinkDesc    - virtual_binding:
        capability: tosca.capabilities.nfv.VirtualBindable
        relationship: tosca.relationships.nfv.VirtualBindsTo
              node: tosca.nodes.nfv.VDU
```

Editor's note: It is for further study whether the requirements should express in the VduCpd or in the Cpd?

5.9.6 tosca.nodes.nfv.VnfVirtualLinkDesc

The TOSCA nfv.VnfVirtualLinkDesc node type represents a logical internal virtual link as defined by [ETSI GS NFV-IFA 011].

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

5.9.6.1 Properties

Name	Required	Type	Constraints	Description
connectivity_type	yes	ConnectivityType		specifies the protocol exposed by the VL and the flow pattern supported by the VL
description	no	string		provides human-readable information on the purpose of the VL (e.g. control plane traffic)
test_access	no	string		Test access facilities available on the VL (e.g. none, passive, monitoring, or active (intrusive) loopbacks at endpoints
vl_flavours	yes	Map of tosca.datatypes.nfv.VlFlavour Editor's note: TBD		Describe a specific flavour of the VL with specific bitrate requirements.

5.9.6.2 Attributes

None

5.9.6.3 Requirements

None

5.9.6.4 Capabilities

Name	Type	Constraints	Description
virtual_linkable	tosca.capabilities.nfv.VirtualLinkable		Defines ability of VirtualLinkable
monitoring_parameter	tosca.capabilities.nfv.Metric Editor's note: TBD	None	Monitoring parameter, which can be tracked for virtualized resource on VL level

5.9.6.5 Definition

```
tosca.nodes.nfv.VnfVirtualLinkDesc:
  derived_from: tosca.nodes.Root
  properties:
    connectivity_type:
      type: tosca.datatypes.nfv.ConnectivityType
      required: true
    description:
      type: string
      required: false
    test_access:
      type: list
      entry_schema:
        type: string
      required: false
    vl_flavours:
      type: map
      entry_schema:
        type: tosca.datatypes.nfv.VlFlavour
      required: true
  capabilities:
    #monitoring_parameters:
      # modeled as ad hoc (named) capabilities in node template
    virtual_linkable:
      type: tosca.capabilities.nfv.VirtualLinkable
```

5.9.6.6 Additional Requirement

5.10 Group Types

5.11 Policy Types

5.12 Using Service Template for a VNFD

5.12.1 Metadata keynames

The following table defines the list of recognized metadata keynames that SHALL be used for NFV VNFD service template:

Keyname	Required	Type	Description

6 Examples

6.1 VNFD modeling design example by using TOSCA composition

The following sample defines a VNFD example which contains three different types of VDUs, interconnected by two virtual link descriptors. In this example, the type of VDU C is not defined within the same VNFD service template file, instead, it is defined in a separate service template file.

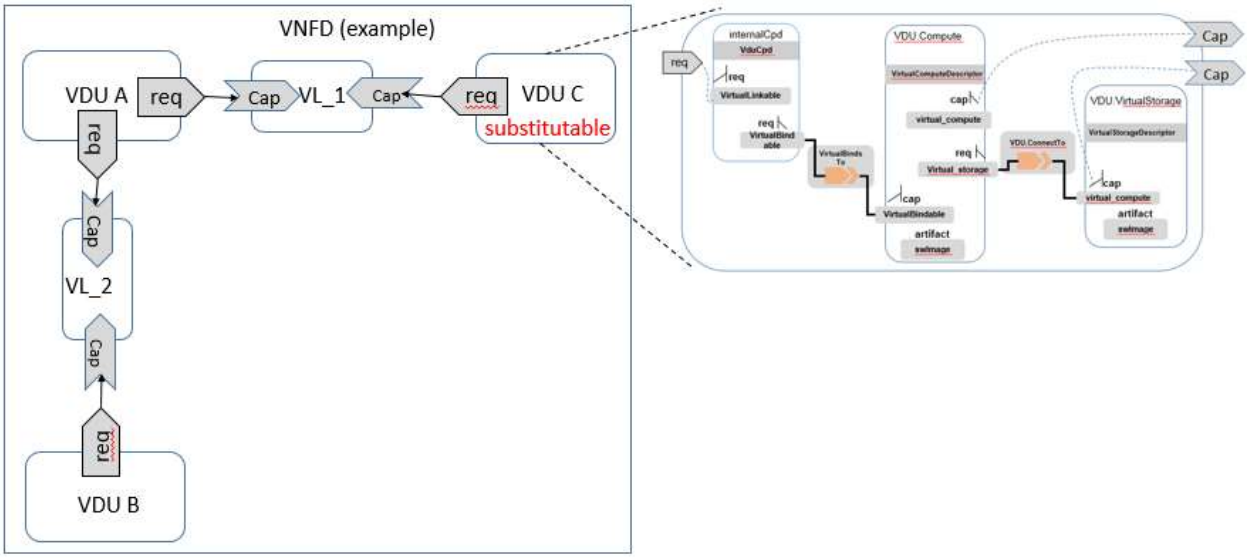


Figure 6.1-1 example of VDU composition design in a VNFD

The service template example of the above VNFD is showing as follow:

```
tosca_definitions_version: tosca_simple_yaml_1_0

topology_template:
  description: Template of a VNFD example

  node_templates:
    VDU_A:
      type: tosca.nodes.nfv.VDUComposition.vduA
      properties:
        # omitted here for brevity
      requirements:
        - virtual_link:VL_1
        - virtual_link:VL_2

    VDU_B:
      type: tosca.nodes.nfv.VDUComposition.vduB
```

```
    properties:
      # omitted here for brevity
    requirements:
      - virtual_link:VL_2

  VDU_C:
    # it can be substituted with a topology provided by another template
     # that exports a virtual_link type’s requirement.
    type: tosca.nodes.nfv. VDUComposition.vduC
    properties:
      # omitted here for brevity
    requirements:
      - virtual_link:VL_1

  VL_1:
    type: tosca.nodes.nfv.VnfVirtualLinkDesc
    properties:
      # omitted here for brevity
    capabilities:
      - virtual_link

  VL_2:
    type: tosca.nodes.nfv.VnfVirtualLinkDesc
    properties:
      # omitted here for brevity
    capabilities:
      - virtual_link
```

The service template example for VDU C is showing as follow.

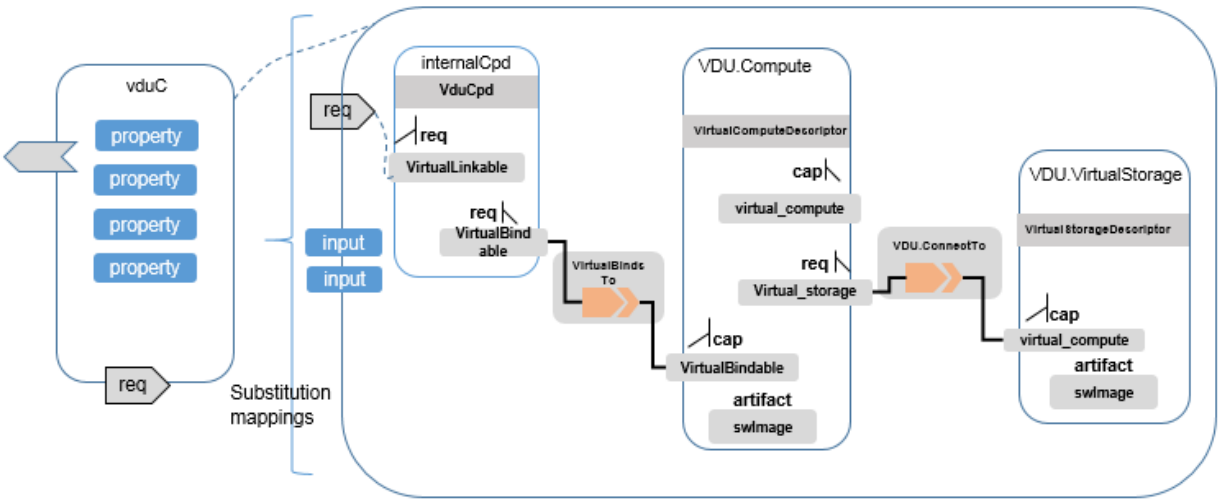


Figure 6.1-2 example of VDU substitution mappings

```
tosca_definitions _version:      tosca_simple_profile_for_nfv_1_0

description: service template of a VDU   # Human readable description of the Vdu.
                    # Human readable name of the Vdu.
topology_template:

  inputs:

  subsititution_mappings:
     node_type: tosca.nodes.nfv.VNF. VDUComposition.vduC  # this is just an example,
                                          users can define their own vdu node type
                                          based on their application
     requirements:
        virtualLinkable: [interanlCpd_001, virtualLinkable]
     capabilities:
        virtual_compute: [vduC_compute, virtual_compute]
        virtual_storage: [vduC_storage, virtual_storage]

  node_templates:
      vduC_compute:   #editor's note: call it VirtualComputeDescriptor or VDU_compute
        type: tosca.nodes.nfv.VDU.Compute
        properties:
           # omitted here for brevity
        capabilities:
          virtual_compute:
        artifacts:
         - sw_image:
```

```
    requirements:
      - virtual_storage: virtualStorage_001

  vduC_storage:    # editor’s note: call it VirtualStorageDescriptor or
                   # VDU_storage
    type: tosca.nodes.nfv.VDU.VirtualStorage
    properties:
       # omitted here for brevity
     capabilities:
      virtual_storage:

  internalCpd:  #ID of this internalCpd
    type: tosca.nodes.nfv.VduCpd
    properties:
       # omitted here for brevity
    requirements:
      - VirtualLinkable:
      - VirtualBindable: vduC_compute
```

The subsititution_mappings section in the above example denotes that this service template can be used for substituting node tempates of type tosca.nodes.nfv.VNF.vduC. The virtualLinkable requirement of internalCpd is exposed as the external requirement of VDU C, which can be used to connect to the VL_1 as showed in figure 4.x.

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 Technologies Co.,Ltd.

Lawrence Lamers (ljlamers@vmware.com), VMware
Sridhar Ramaswamy (sramasw@Brocade.com), Brocade

John Crandall (jcrandal@Brocade.COM), Brocade

Thinh Nguyenphu (thinh.nguyenphu@nokia.com), Nokia

Dmytro Gassanov (dmytro.gassanov@NetCracker.com), NetCracker

Andrei Chekalin (chekalin@netcracker.com), NetCracker

Preetdeep Kumar (preetdeep.kumar@ca.com), CA Technologies

Bruce Thompson (brucet@cisco.com), Cisco Systems

Steve Baillargeon (steve.baillargeon@ericsson.com), Ericsson

Alexander Vul (alex.vul@intel.com), Intel Corporation

Michael Brenner (michael@gigaspaces.com),GigaSpaces Technologies

Hui Deng (denghui12@huawei.com), Huawei Technologies Co.,Ltd.

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
WD02,Rev01	2015-7-2	Shitao li, Huawei	• 6.1, changing the version number from 1.0.0 to 1.0 • 6.2, adding NFV usage specific metadata keynames

			• 6.3, using metadata element instead of service_properties • 7.1, using metadata element instead of service_properties
WD02,Rev02	2015-8-26	Shitao li, Huawei	• 6: change title to “TOSCA Data model for a network service”, and move the NSD example as well as NSD related definition to clause 11. • 7: change title to “TOSCA Data model for a VNF” • 8.1: in the text and the VNFD example, adding Forwarder capability to exteral connection point for supporting NFP description • 10: moving VNFFG description text from clause 3.3 to clause 10. • 10.1,10.2,10.3,10.4,10.5,10.6: adding TOSCA model for VNFFG, using group type for VNFFG and node type for NFP • 11: moving TOSCA template for NSD from clause 7 to clause 11. • 11.2: adding VNFFG and NFP in the NSD example
WD02, Rew03	2015-9-28	Matt Rutkowski, IBM	• 11.2: changing NSD example for NFP, adding “-” in front of every requirement.
WD02, Rew04	2015-10-15	Chris Lauwers	• Formatting changes
WD02, Rew05	2016-1-22	Sridhar Ramaswamy, Brocade Shitao li, Huawei	• 12, adding new VNFD example for the single vRouter use case.
WD02, Rev07	2016-2-18	Sridhar Ramaswamy, Brocade Matt Rutkowski, IBM	• 13. Enhance VDU with CPU Architecture properties like CPU pinning, Huge-pages, NUMA topology, etc. • 13.2 Change, VirtualLink, ConnectionPoint to derive from / use appropriate Simple YAML Profile node_types and datatypes.
WD02, Rev08	2016-2-25	Sridhar Ramaswamy, Brocade	• Add anti-spoof protection flag to ConnectionPoint • Update the samples based on new CPU Architecture Schema • Add NFV Profile sample with efficient CPU and Memory allocation • Add NFV profile sample with multiple VDUs
WD02, Rev09	2016-2-29	Sridhar Ramaswamy, Brocade	• Move Compute Architecture capability and related datatypes to Sec 8. • Add diagram for multi-vdu VNFD template example • Add a note on artifacts for VDU
WD03, Rev01	2016-7-29	Shitao Li Huawei	• Solve Issue TOSCA-289: Invalid definition for tosca.capabilities.Compute.Container.Architecture • Solve Issue TOSCA-291: Invalid definition of tosca.nodes.nfv.VL.ELine • Solve Issue TOSCA-293: tosca.nodes.nfv.CP type has "IP_address" as an attribute • Solve Issue TOSCA-294: Inconsistent usage of anti_spoofing_protection CP property • Solve Issue TOSCA-304: [TOSCA-Simple-Profile-NFV-v1.0] csd03 references an out of date ETSI specification • Solve Issue TOSCA-310: Adding vEPC NSD example
WD04,	2016-	Shitao Li,	• Solve Issue TOSCA-305: Proposal modification to ToC based on document

Rev02	9-6	Huawei	Issue_TOSCA305_tosca-nfv-v1.0-wd03-rev01 TOC_r3 ● Solve Issue TOSCA-311: Adding vEPC NSD example
WD04, Rev03	2016-11-7	Shitao Li, Huawei	● Adding new data types for connection point and virtual link based on ETSI NFV IFA011 ● Moving ETSI GS NFV-MAN 001 v1.1.1 into informative reference. ● Solve Issue TOSCA-307 and TOSCA-308: adding new node type Cpd, VduCpd and VnfVirtualLinkDesc
WD04, Rev04	2016-11-14	Shitao Li, Huawei	● Editorial changes based on document Issue_TOSCA307_ConnectionPoint_VL_change proposal
WD04, Rev05	2017-1-17	Shitao Li, Huawei	● Editorial changes for Cpd and VnfVirtualLinkDesc based on document Issue_TOSCA307_ConnectionPoint_VL_change proposal-r2 ● Clause 5.9.5.5, changed tosca.nodes.nfv.VL to tosca.nodes.nfv.VnfVirtualLinkDesc to align with IFA011. ● Deleted the legacy contents which are not aligned with IFA011: • clause 5.1 • clause 5.9.1, the node type definition of VNF • clause 5.9.2, the node type definition of VDU. • examples of VNFD and NSD • tosca.datatypes.compute.Container.Architecture.CPUAllocation • tosca.datatypes.compute.Container.Architecture.NUMA • tosca.capabilities.Compute.Container.Architecture ● Deleted NSD related content, v1.0 will only cover VNFD model. ● Removed ETSI GS NFV-MAN 001 v1.1.1 in the reference. ● Added text in 5.1 and 5.2 based on document Issue_TOSCA306_VNFD_IE_to_TOSCA_Types r5
WD04, Rev06	2017-2-15	Shitao Li, Huawei	● Added VDU.Compute definition in document “Issue_TOSCA310_VDU change proposal-option3a draft3” ● Added VNFD metadata keynames in document “VNFD metadata discussion”
WD04, Rev07	2017-4-12	Shitao Li, Huawei	● Added VDU design by using TOSCA composition in 4.3 ● Added an example of VNFD modelling design in 6.1 ● Deleted Metadata keynames definition in 5.12.1
WD04, Rev08	2017-4-26	Shitao Li, Huawei	● Modification and clarification on general description of VNFD and NSD in clause 3.
WD04, Rev09	2017-5-4	Shitao Li, Huawei	● Clarification of vdu.compute in 5.9.2

