



Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference

DGS/NFV-SOL001

Keywords

data, information model, model, NFV

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:

<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:

<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2018.

All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.

3GPP™ and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

oneM2M™ logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners

GSM® and the GSM logo are trademarks registered and owned by the GSM Association.

Contents

Intellectual Property Rights	15
Foreword.....	15
Modal verbs terminology.....	15
1 Scope	16
2 References	16
2.1 Normative references	16
2.2 Informative references.....	17
3 Definition of terms and abbreviations	17
3.1 Terms.....	17
3.2 Abbreviations	17
4 Overview of TOSCA model.....	18
5 General concept of using TOSCA to model NFV descriptors	18
5.1 Introduction	18
5.2 Network Service Descriptor	18
5.3 Virtualised Network Function Descriptor	19
5.4 toasca_definitions_version and Namespace prefix	19
5.5 Imports statement	20
5.5.1 VNFD TOSCA service template	20
5.5.2 NSD TOSCA service template	20
5.5.3 PNFD TOSCA service template	21
5.6 Type extension	21
5.6.1 Introduction.....	21
5.6.2 Rules	22
5.6.3 VNFD Types.....	22
5.6.4 NSD types.....	22
6 VNFD TOSCA model.....	23
6.1 Introduction	23
6.2 Data Types.....	24
6.2.1 toasca.datatypes.nfv.L2AddressData.....	24
6.2.1.1 Description	24
6.2.1.2 Properties	24
6.2.1.3 Definition	24
6.2.1.4 Examples.....	25
6.2.1.5 Additional Requirements	25
6.2.2 toasca.datatypes.nfv.L3AddressData.....	25
6.2.2.1 Description	25
6.2.2.2 Properties	25
6.2.2.3 Definition	26
6.2.2.4 Examples.....	26
6.2.2.5 Additional Requirements	26
6.2.3 toasca.datatypes.nfv.AddressData	26
6.2.3.1 Description	26
6.2.3.2 Properties	27
6.2.3.3 Definition	27
6.2.3.4 Examples.....	27
6.2.3.5 Additional Requirements	28
6.2.4 toasca.datatypes.nfv.VirtualNetworkInterfaceRequirements	28
6.2.4.1 Description	28
6.2.4.2 Properties	28
6.2.4.3 Definition	29
6.2.4.4 Examples.....	29
6.2.4.5 Additional Requirements	29
6.2.5 toasca.datatypes.nfv.ConnectivityType	29

6.2.5.1	Description	29
6.2.5.2	Properties	30
6.2.5.3	Definition	30
6.2.5.4	Examples	30
6.2.5.5	Additional Requirements	30
6.2.6	tosca.datatypes.nfv.RequestedAdditionalCapability	31
6.2.6.1	Description	31
6.2.6.2	Properties	31
6.2.6.3	Definition	31
6.2.6.4	Examples	32
6.2.6.5	Additional Requirements	32
6.2.7	tosca.datatypes.nfv.VirtualMemory	32
6.2.7.1	Description	32
6.2.7.2	Properties	32
6.2.7.3	Definitions	33
6.2.7.4	Examples	33
6.2.7.5	Additional Requirements	33
6.2.8	tosca.datatypes.nfv.VirtualCpu	33
6.2.8.1	Description	33
6.2.8.2	Properties	33
6.2.8.3	Definition	34
6.2.8.4	Examples	35
6.2.8.5	Additional Requirements	35
6.2.9	tosca.datatypes.nfv.VirtualCpuPinning	35
6.2.9.1	Description	35
6.2.9.2	Properties	35
6.2.9.3	Definition	35
6.2.9.4	Examples	36
6.2.9.5	Additional Requirements	36
6.2.10	tosca.datatypes.nfv.VnfcConfigurableProperties	36
6.2.10.1	Description	36
6.2.10.2	Properties	36
6.2.10.3	Definition	37
6.2.10.4	Examples	37
6.2.10.5	Additional Requirements	37
6.2.11	tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties	38
6.2.11.1	Description	38
6.2.11.2	Properties	38
6.2.11.3	Definition	38
6.2.12	tosca.datatypes.nfv.VduProfile	38
6.2.12.1	Description	38
6.2.12.2	Properties	38
6.2.12.3	Definition	39
6.2.12.4	Examples	39
6.2.12.5	Additional requirements	40
6.2.13	tosca.datatypes.nfv.VIProfile	41
6.2.13.1	Description	41
6.2.13.2	Properties	41
6.2.13.3	Definition	41
6.2.13.4	Examples	42
6.2.13.5	Additional Requirements	42
6.2.14	tosca.datatypes.nfv.VirtualLinkProtocolData	42
6.2.14.1	Description	42
6.2.14.2	Properties	42
6.2.14.3	Definition	42
6.2.14.4	Examples	43
6.2.14.5	Additional Requirements	43
6.2.15	tosca.datatypes.nfv.L2ProtocolData	43
6.2.15.1	Description	43
6.2.15.2	Properties	43
6.2.15.3	Definition	44
6.2.15.4	Examples	44

6.2.15.5	Additional Requirements	44
6.2.16	tosca.datatypes.nfv.L3ProtocolData	45
6.2.16.1	Description	45
6.2.16.2	Properties	45
6.2.16.3	Definition	45
6.2.16.4	Examples	46
6.2.16.5	Additional Requirements	46
6.2.17	tosca.datatypes.nfv.IpAllocationPool	46
6.2.17.1	Description	46
6.2.17.2	Properties	47
6.2.17.3	Definition	47
6.2.17.4	Examples	47
6.2.17.5	Additional Requirements	47
6.2.18	tosca.datatypes.nfv.InstantiationLevel	47
6.2.18.1	Description	47
6.2.18.2	Properties	48
6.2.18.3	Definition	48
6.2.18.4	Examples	48
6.2.18.5	Additional Requirements	48
6.2.19	tosca.datatypes.nfv.VduLevel	48
6.2.19.1	Description	48
6.2.19.2	Properties	49
6.2.19.3	Definition	49
6.2.19.4	Examples	49
6.2.19.5	Additional Requirements	49
6.2.20	tosca.datatypes.nfv.VnfLcmOperationsConfiguration	49
6.2.20.1	Description	49
6.2.20.2	Properties	50
6.2.20.3	Definition	50
6.2.20.4	Examples	51
6.2.20.5	Additional Requirements	51
6.2.21	tosca.datatypes.nfv.VnfInstantiateOperationConfiguration	51
6.2.21.1	Description	51
6.2.21.2	Properties	51
6.2.21.3	Definition	51
6.2.21.4	Examples	52
6.2.21.5	Additional Requirements	52
6.2.22	tosca.datatypes.nfv.VnfScaleOperationConfiguration	52
6.2.22.1	Description	52
6.2.22.2	Properties	52
6.2.22.3	Definition	52
6.2.22.4	Examples	52
6.2.22.5	Additional Requirements	53
6.2.23	tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration	53
6.2.23.1	Description	53
6.2.23.2	Properties	53
6.2.23.3	Definition	53
6.2.23.4	Examples	53
6.2.23.5	Additional Requirements	53
6.2.24	tosca.datatypes.nfv.VnfHealOperationConfiguration	54
6.2.24.1	Description	54
6.2.24.2	Properties	54
6.2.24.3	Definition	54
6.2.24.4	Examples	54
6.2.24.5	Additional Requirements	54
6.2.25	tosca.datatypes.nfv.VnfTerminateOperationConfiguration	54
6.2.25.1	Description	54
6.2.25.2	Properties	55
6.2.25.3	Definition	55
6.2.25.4	Examples	55
6.2.25.5	Additional Requirements	55
6.2.26	tosca.datatypes.nfv.VnfOperateOperationConfiguration	56

6.2.26.1	Description	56
6.2.26.2	Properties	56
6.2.26.3	Definition	56
6.2.26.4	Examples	56
6.2.26.5	Additional Requirements	57
6.2.27	tosca.datatypes.nfv.ScaleInfo	57
6.2.27.1	Description	57
6.2.27.2	Properties	57
6.2.27.3	Definition	57
6.2.27.4	Examples	57
6.2.27.5	Additional Requirements	57
6.2.28	tosca.datatypes.nfv.ScalingAspect	57
6.2.28.1	Description	57
6.2.28.2	Properties	58
6.2.28.3	Definition	58
6.2.28.4	Examples	58
6.2.28.5	Additional Requirements	59
6.2.29	tosca.datatypes.nfv.LinkBitrateRequirements	59
6.2.29.1	Description	59
6.2.29.2	Properties	59
6.2.29.3	Definition	59
6.2.29.4	Examples	60
6.2.29.5	Additional Requirements	60
6.2.30	tosca.datatypes.nfv.Qos	60
6.2.30.1	Description	60
6.2.30.2	Properties	60
6.2.30.3	Definition	60
6.2.30.4	Examples	61
6.2.30.5	Additional Requirements	61
6.2.31	tosca.datatypes.nfv.VnfConfigurableProperties	61
6.2.31.1	Description	61
6.2.31.2	Properties	61
6.2.31.3	Definition	61
6.2.31.4	Examples	62
6.2.31.5	Additional Requirements	63
6.2.32	tosca.datatypes.nfv.VnfAdditionalConfigurableProperties	63
6.2.32.1	Description	63
6.2.32.2	Properties	63
6.2.32.3	Definition	63
6.2.32.4	Examples	63
6.2.32.5	Additional Requirements	63
6.2.33	tosca.datatypes.nfv.VnfInfoModifiableAttributes	63
6.2.33.1	Description	63
6.2.33.2	Properties	63
6.2.33.3	Definition	64
6.2.33.4	Examples	64
6.2.33.5	Additional Requirements	64
6.2.34	tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions	64
6.2.34.2	Properties	65
6.2.34.3	Definition	65
6.2.34.4	Examples	65
6.2.34.5	Additional Requirements	65
6.2.35	tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata	65
6.2.35.1	Description	65
6.2.35.2	Properties	65
6.2.35.3	Definition	65
6.2.35.4	Examples	65
6.2.35.5	Additional Requirements	66
6.2.36	tosca.datatypes.nfv.CpProtocolData	66
6.2.36.1	Description	66
6.2.36.2	Properties	66
6.2.36.3	Definition	67

6.2.36.4	Examples	67
6.2.36.5	Additional Requirements	67
6.2.37	tosca.datatypes.nfv.LogicalNodeData	67
6.2.37.1	Description	67
6.2.37.2	Properties	68
6.2.37.3	Definition	68
6.2.37.4	Examples	68
6.2.37.5	Additional Requirements	68
6.2.38	tosca.datatypes.nfv.SwImageData	68
6.2.38.1	Description	68
6.2.38.2	Properties	69
6.2.38.3	Definition	70
6.2.38.4	Examples	71
6.2.38.5	Additional Requirements	71
6.2.39	tosca.datatypes.nfv.VirtualBlockStorageData	71
6.2.39.1	Description	71
6.2.39.2	Properties	71
6.2.39.3	Definition	71
6.2.39.4	Examples	72
6.2.39.5	Additional Requirements	72
6.2.40	tosca.datatypes.nfv.VirtualObjectStorageData	72
6.2.40.1	Description	72
6.2.40.2	Properties	72
6.2.40.3	Definition	72
6.2.40.4	Examples	73
6.2.40.5	Additional Requirements	73
6.2.41	tosca.datatypes.nfv.VirtualFileStorageData	73
6.2.41.1	Description	73
6.2.41.2	Properties	73
6.2.41.3	Definition	73
6.2.41.4	Examples	74
6.2.41.5	Additional Requirements	74
6.2.42	tosca.datatypes.nfv.VirtualLinkBitrateLevel	74
6.2.42.1	Description	74
6.2.42.2	Properties	74
6.2.42.3	Definition	74
6.2.42.4	Examples	74
6.2.42.5	Additional Requirements	74
6.2.43	tosca.datatypes.nfv.VnfOperationAdditionalParameters	75
6.2.43.1	Description	75
6.2.43.2	Properties	75
6.2.43.3	Definition	75
6.2.43.4	Examples	75
6.2.43.5	Additional Requirements	76
6.2.44	tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration	76
6.2.44.1	Description	76
6.2.44.2	Properties	76
6.2.44.3	Definition	76
6.2.44.4	Examples	76
6.2.44.5	Additional Requirements	76
6.2.45	tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration	76
6.2.45.1	Description	76
6.2.45.2	Properties	77
6.2.45.3	Definition	77
6.2.45.4	Examples	77
6.2.45.5	Additional Requirements	77
6.2.46	tosca.datatypes.nfv.VnfMonitoringParameter	77
6.2.46.1	Description	77
6.2.46.2	Properties	77
6.2.46.3	Definition	78
6.2.46.4	Examples	79
6.2.46.5	Additional Requirements	79

6.2.47	tosca.datatypes.nfv.VnfcMonitoringParameter.....	79
6.2.47.1	Description	79
6.2.47.2	Properties	79
6.2.47.3	Definition	80
6.2.47.4	Examples.....	81
6.2.47.5	Additional Requirements	81
6.2.48	tosca.datatypes.nfv.VirtualLinkMonitoringParameter	81
6.2.48.1	Description	81
6.2.48.2	Properties	81
6.2.48.3	Definition	82
6.2.48.4	Examples.....	82
6.2.48.5	Additional Requirements	82
6.2.49	tosca.datatypes.nfv.InterfaceDetails	82
6.2.49.1	Description	82
6.2.49.2	Properties	82
6.2.49.3	Definition	83
6.2.49.4	Examples.....	83
6.2.49.5	Additional Requirements	83
6.2.50	tosca.datatypes.nfv.UriComponents	83
6.2.50.1	Description	83
6.2.50.2	Properties	83
6.2.50.3	Definition	84
6.2.50.4	Examples.....	84
6.2.50.5	Additional Requirements	85
6.2.51	tosca.datatypes.nfv.UriAuthority	85
6.2.51.1	Description	85
6.2.51.2	Properties	85
6.2.51.3	Definition	85
6.2.51.4	Examples.....	86
6.2.51.5	Additional Requirements	86
6.2.52	tosca.datatypes.nfv.VnfProfile.....	86
6.2.52.1	Description	86
6.2.52.2	Properties	86
6.2.52.3	Definition	87
6.2.52.4	Example	87
6.2.52.5	Additional Requirements	87
6.3	Artifact Types.....	88
6.3.1	tosca.artifacts.nfv.SwImage	88
6.3.1.1	Description	88
6.3.1.2	Description	88
6.3.2	tosca.artifacts.Implementation.nfv.Mistral	88
6.3.2.1	Description	88
6.3.2.2	Definition	88
6.4	Capability Types	89
6.4.1	tosca.capabilities.nfv.VirtualBindable	89
6.4.1.1	Description	89
6.4.1.2	Properties	89
6.4.1.3	Definition	89
6.4.2	tosca.capabilities.nfv.VirtualLinkable	89
6.4.2.1	Description	89
6.4.2.2	Properties	89
6.4.2.3	Definition	90
6.4.3	tosca.capabilities.nfv.VirtualCompute.....	90
6.4.3.1	Description	90
6.4.3.2	Properties	90
6.4.3.3	Definition	90
6.4.4	tosca.capabilities.nfv.VirtualStorage	91
6.4.4.1	Description	91
6.4.4.2	Definition	91
6.5	Requirements Types	92
6.6	Relationship Types	92
6.6.1	tosca.relationships.nfv.VirtualBindsTo.....	92

6.6.1.1	Description	92
6.6.1.2	Properties	92
6.6.1.3	Definition	92
6.6.2	tosca.relationships.nfv.VirtualLinksTo	92
6.6.2.1	Description	92
6.6.2.2	Properties	92
6.6.2.3	Definition	93
6.6.3	tosca.relationships.nfv.AttachesTo	93
6.6.3.1	Description	93
6.6.3.2	Properties	93
6.6.3.3	Definition	93
6.7	Interface Types	93
6.7.1	tosca.interfaces.nfv.Vnflcm	93
6.7.1.1	Description	93
6.7.1.2	Definition	94
6.7.1.3	Additional Requirements	96
6.7.1.4	Support of LCM scripts.....	96
6.7.1.5	Examples	97
6.8	Node Types	99
6.8.1	tosca.nodes.nfv.VNF.....	99
6.8.1.1	Description	99
6.8.1.2	Properties	100
6.8.1.3	Attributes.....	102
6.8.1.4	Requirements	102
6.8.1.5	Capabilities.....	102
6.8.1.6	Definition	102
6.8.1.7	Artifact	104
6.8.1.8	Additional Requirements	104
6.8.1.9	Example	105
6.8.2	tosca.nodes.nfv.VnfExtCp	106
6.8.2.1	Description	106
6.8.2.2	Properties	106
6.8.2.3	Attributes.....	107
6.8.2.4	Requirements	107
6.8.2.5	Definition	107
6.8.2.6	Additional Requirements	107
6.8.2.7	Example	108
6.8.3	tosca.nodes.nfv.Vdu.Compute	108
6.8.3.1	Description	108
6.8.3.2	Properties	108
6.8.3.3	Attributes.....	109
6.8.3.4	Requirements	109
6.8.3.5	Capabilities.....	109
6.8.3.6	Definition	110
6.8.3.7	Additional requirements.....	111
6.8.3.8	Example	111
6.8.4	tosca.nodes.nfv.Vdu.VirtualBlockStorage	113
6.8.4.1	Description	113
6.8.4.2	Properties	113
6.8.4.3	Attributes.....	113
6.8.4.4	Requirements	113
6.8.4.5	Capabilities.....	113
6.8.4.6	Definition	113
6.8.4.7	Additional requirements.....	114
6.8.5	tosca.nodes.nfv.Vdu.VirtualObjectStorage.....	114
6.8.5.1	Description	114
6.8.5.2	Properties	114
6.8.5.3	Attributes.....	114
6.8.5.4	Requirements	114
6.8.5.5	Capabilities.....	114
6.8.5.6	Definition	115
6.8.5.7	Additional requirements.....	115

6.8.6	tosca.nodes.nfv.Vdu.VirtualFileStorage	115
6.8.6.1	Description	115
6.8.6.2	Properties	115
6.8.6.3	Attributes	115
6.8.6.4	Requirements	116
6.8.6.5	Capabilities	116
6.8.6.6	Definition	116
6.8.6.7	Additional requirements	116
6.8.7	tosca.nodes.nfv.Cp	116
6.8.7.1	Description	116
6.8.7.2	Properties	117
6.8.7.3	Attributes	117
6.8.7.4	Requirements	117
6.8.7.5	Capabilities	117
6.8.7.6	Definition	118
6.8.7.7	Additional requirements	118
6.8.8	tosca.nodes.nfv.VduCp	118
6.8.8.1	Description	118
6.8.8.2	Properties	119
6.8.8.3	Attributes	119
6.8.8.4	Requirements	119
6.8.8.5	Definition	120
6.8.9	tosca.nodes.nfv.VnfVirtualLink	120
6.8.9.1	Description	120
6.8.9.2	Properties	121
6.8.9.3	Requirements	121
6.8.9.4	Capabilities	121
6.8.9.5	Definition	121
6.9	Group Types	122
6.9.1	tosca.groups.nfv.PlacementGroup	122
6.9.1.1	Description	122
6.9.1.2	Properties	122
6.9.1.3	Definition	122
6.9.1.4	Additional Requirements	123
6.9.1.5	Examples	123
6.10	Policy Types	123
6.10.1	tosca.policies.nfv.InstantiationLevels	123
6.10.1.1	Description	123
6.10.1.2	Properties	123
6.10.1.3	Definition	123
6.10.2	tosca.policies.nfv.VduInstantiationLevels	124
6.10.2.1	Description	124
6.10.2.2	Properties	124
6.10.2.3	Definition	124
6.10.2.4	Additional Requirements	125
6.10.3	tosca.policies.nfv.VirtualLinkInstantiationLevels	125
6.10.3.1	Description	125
6.10.3.2	Properties	125
6.10.3.3	Definition	125
6.10.3.4	Additional Requirements	126
6.10.4	Void	126
6.10.5	tosca.policies.nfv.ScalingAspects	126
6.10.5.1	Description	126
6.10.5.2	Properties	126
6.10.5.3	Definition	126
6.10.5.4	Examples	127
6.10.6	tosca.policies.nfv.VduScalingAspectDeltas	127
6.10.6.1	Description	127
6.10.6.2	Properties	127
6.10.6.3	Definition	127
6.10.6.4	Examples	127
6.10.7	tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas	128

6.10.7.1	Description	128
6.10.7.2	Properties	128
6.10.7.3	Definition	128
6.10.7.4	Examples	128
6.10.8	tosca.policies.nfv.VduInitialDelta	129
6.10.8.1	Description	129
6.10.8.2	Properties	129
6.10.8.3	Definition	129
6.10.8.4	Examples	129
6.10.9	tosca.policies.nfv.VirtualLinkBitrateInitialDelta	129
6.10.9.1	Description	129
6.10.9.2	Properties	130
6.10.9.3	Definition	130
6.10.9.4	Examples	130
6.10.10	AffinityRule, AntiAffinityRule	130
6.10.10.1	Description	130
6.10.10.2	Properties	131
6.10.10.3	targets	131
6.10.10.4	Definition	131
6.10.10.5	Examples	132
6.10.11	tosca.policies.nfv.SecurityGroupRule	133
6.10.11.1	Description	133
6.10.11.2	Properties	133
6.10.11.3	Definition	134
6.10.11.4	Additional Requirements	135
6.10.12	tosca.policies.nfv.SupportedVnfInterface	135
6.10.12.1	Description	135
6.10.12.2	Properties	135
6.10.12.3	Definition	135
6.10.12.4	Additional requirements	136
6.10.12.5	Example	136
6.11	VNFD TOSCA service template design	136
6.11.1	General	136
6.11.2	Single or multiple deployment flavour design with TOSCA-Simple-Profile-YAML-v1.2	137
6.11.3	Single deployment flavour design with TOSCA-Simple-Profile-YAML-v1.1	138
7	NSD TOSCA model	138
7.1	Introduction	138
7.2	Data Types	139
7.2.1	tosca.datatype.nfv.ServiceAvailability	139
7.2.1.1	Description	139
7.2.1.2	Properties	139
7.2.1.3	Definition	139
7.2.1.4	Examples	140
7.2.2	tosca.datatypes.nfv.VnfProfile	140
7.2.2.1	Description	140
7.2.3	tosca.datatype.nfv.NsVIProfile	140
7.2.3.1	Description	140
7.2.3.2	Properties	140
7.2.3.3	Definition	140
7.2.3.4	Examples	141
7.2.3.5	Additional Requirements	141
7.2.4	tosca.datatypes.nfv.ConnectivityType	141
7.2.4.1	Description	141
7.2.5	tosca.datatypes.nfv.NsVirtualLinkQos	141
7.2.5.1	Description	141
7.2.5.2	Properties	141
7.2.5.3	Definition	142
7.2.5.4	Examples	142
7.2.5.5	Additional Requirements	142
7.2.6	tosca.datatypes.nfv.LinkBitrateRequirements	142
7.2.6.1	Description	142

7.2.7	tosca.datatypes.nfv.VirtualLinkProtocolData	142
7.2.8	tosca.datatypes.nfv.L2ProtocolData	142
7.2.8.1	Description	142
7.2.9	tosca.datatypes.nfv.L3ProtocolData	142
7.2.9.1	Description	142
7.2.10	tosca.datatypes.nfv.IpAllocationPool	142
7.2.10.1	Description	142
7.2.11	tosca.datatypes.nfv.CpProtocolData	143
7.2.11.1	Description	143
7.2.12	tosca.datatypes.nfv.AddressData	143
7.2.12.1	Description	143
7.2.13	tosca.datatypes.nfv.L2AddressData	143
7.2.13.1	Description	143
7.2.14	tosca.datatypes.nfv.L3AddressData	143
7.2.14.1	Description	143
7.3	Artifact Types	143
7.4	Capabilities Types	143
7.4.1	tosca.capabilities.nfv.VirtualLinkable	143
7.4.1.1	Description	143
7.5	Requirements Types	143
7.6	Relationship Types	144
7.6.1	tosca.relationships.nfv.VirtualLinksTo	144
7.6.1.1	Description	144
7.7	Interface Types	144
7.7.1	tosca.interfaces.nfv.Nslcm	144
7.7.1.1	Description	144
7.7.1.2	Definition	144
7.7.1.3	Additional Requirements	145
7.7.1.4	Support of LCM scripts	145
7.7.1.5	Examples	146
7.8	Node Types	147
7.8.1	tosca.nodes.nfv.NS	147
7.8.1.1	Description	147
7.8.1.2	Properties	147
7.8.1.3	Attributes	147
7.8.1.4	Requirements	147
7.8.1.5	Capabilities	147
7.8.1.6	Definition	148
7.8.1.7	Artifact	148
7.8.1.8	Additional requirements	148
7.8.2	tosca.nodes.nfv.Sap	149
7.8.2.1	Description	149
7.8.2.2	Properties	149
7.8.2.3	Attributes	149
7.8.2.4	Requirements	149
7.8.2.5	Capabilities	149
7.8.2.6	Definition	149
7.8.2.7	Additional requirements	150
7.8.2.8	Example	150
7.8.3	tosca.nodes.nfv.NsVirtualLink	150
7.8.3.1	Description	150
7.8.3.2	Properties	150
7.8.3.3	Attributes	151
7.8.3.4	Requirements	151
7.8.3.5	Capabilities	151
7.8.3.6	Definition	151
7.8.3.7	Artifact	151
7.8.3.8	Additional Requirements	151
7.8.3.9	Example	152
7.8.4	tosca.nodes.nfv.Cp	152
7.8.4.1	Description	152
7.9	Group Types	152

7.9.1	tosca.groups.nfv.PlacementGroup	152
7.9.1.1	Description	152
7.10	Policy Types	152
7.10.1	AffinityRule, AntiAffinityRule.....	152
7.10.1.1	Description	152
7.10.1.2	Targets.....	152
7.10.1.3	Examples	153
7.10.2	tosca.policies.nfv.SecurityGroupRule.....	153
7.10.2.1	Description	153
7.10.2.2	Additional Requirements	153
8	PNFD TOSCA model.....	154
8.1	Introduction	154
8.2	Data Types.....	154
8.2.1	tosca.datatypes.nfv.CpProtocolData	154
8.2.1.1	Description	154
8.2.2	tosca.datatypes.nfv.AddressData	155
8.2.2.1	Description	155
8.2.3	tosca.datatypes.nfv.L2AddressData.....	155
8.2.3.1	Description	155
8.2.4	tosca.datatypes.nfv.L3AddressData.....	155
8.2.4.1	Description	155
8.2.5	tosca.datatypes.nfv.LocationInfo	155
8.2.5.1	Description	155
8.2.5.2	Properties	155
8.2.5.3	Definition	155
8.2.5.4	Examples.....	156
8.2.5.5	Additional Requirements	156
8.2.6	tosca.datatypes.nfv.CivicAddressElement	156
8.2.6.1	Description	156
8.2.6.2	Properties	156
8.2.6.3	Definition	157
8.2.6.4	Examples.....	157
8.2.6.5	Additional Requirements	157
8.3	Artifact Types.....	157
8.4	Capabilities Types	157
8.4.1	tosca.capabilities.nfv.VirtualLinkable	157
8.4.1.1	Description	157
8.5	Requirements Types	157
8.6	Relationship Types	157
8.6.1	tosca.relationships.nfv.VirtualLinksTo.....	157
8.6.1.1	Description	157
8.7	Interface Types	158
8.8	Node Types	158
8.8.1	tosca.nodes.nfv.PNF	158
8.8.1.1	Description	158
8.8.1.2	Properties	158
8.8.1.3	Attributes.....	158
8.8.1.4	Requirements	158
8.8.1.5	Capabilities.....	159
8.8.1.6	Definition	159
8.8.1.7	Artifact	159
8.8.1.8	Additional Requirements	159
8.8.1.9	Example	160
8.8.2	tosca.nodes.nfv.PnfExtCp.....	160
8.8.2.1	Description	160
8.8.2.2	Properties	160
8.8.2.3	Attributes.....	160
8.8.2.4	Requirements	160
8.8.2.5	Capabilities.....	160
8.8.2.6	Definition	160
8.8.3	tosca.nodes.nfv.Cp.....	161

8.8.3.1	Description	161
8.9	Group Types	161
8.10	Policy Types	161
8.10.1	tosca.policies.nfv.SecurityGroupRule.....	161
8.10.1.1	Description	161
8.10.1.2	Additional Requirements	161
Annex A (informative): Examples.....		162
A.1	Deployment flavour design mapping	162
A.1.1	Introduction	162
A.1.2	Design principle	162
A.2	VNFD with deployment flavour modelling design example.....	163
A.3	VNF external connection point	171
A.3.1	General	171
A.3.2	External connection point re-exposing an internal connection point.....	171
A.3.3	External connection point connected to an internal virtual link	173
A.4	VNFD modeling design example by using TOSCA composition.....	174
A.5	VNFD with Single deployment flavour modelling design example	177
A.6	Scaling and Instantiation Level examples	180
A.6.1	ScalingAspect and InstantiationLevels policies with uniform delta.....	180
A.6.2	ScalingAspect and InstantiationLevels policies with non-uniform deltas	186
A.7	Service Access Point	189
A.7.1	General	189
A.7.2	VNF External connection point exposing as a SAP	190
A.7.3	SAP connected to an NS virtual link	191
A.8	NSD with Single deployment flavour modelling design example	192
A.9	Mapping between NFV IM and TOSCA concepts.....	194
A.9.1	Introduction	194
A.9.2	Mapping between ETSI GS NFV-IFA 011 IM and TOSCA concepts.....	195
A.9.3	Mapping between IFA014 IM and TOSCA concepts	196
A.10	PNFD modelling design example.....	197
Annex B (normative): etsi_nfv_sol001_type definitions		199
B.1	Purpose	199
B.2	VNFD type definitions file.....	199
B.3	NSD type definitions file.....	199
B.4	PNFD type definitions file	199
Annex C (normative): Conformance		200
C.1	Purpose	200
C.2	NFV TOSCA YAML service template	200
C.3	NFV TOSCA processor.....	201
Annex D (informative): Authors & contributors.....		202
Annex E (informative): Change History		203
History		205

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<https://ipr.etsi.org/>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](#) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies a data model for NFV descriptors, using the TOSCA Simple Profile in YAML [3], fulfilling the requirements specified in ETSI GS NFV-IFA 011 [1] and ETSI GS NFV-IFA 014 [2] for a Virtualised Network Function Descriptor (VNFD), a Network Service Descriptor (NSD) and a Physical Network Function Descriptor (PNFD). The present document also specifies requirements on the VNFM and NFVO specific to the handling of NFV descriptors based on the TOSCA Simple Profile in YAML specification [3].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at <https://docbox.etsi.org/Reference/>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

- [1] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Descriptor and Packaging Specification".
- [2] ETSI GS NFV-IFA 014: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification".
- [3] TOSCA-Simple-Profile-yaml-v1.2: "TOSCA Simple Profile in YAML Version 1.2".
- [4] TOSCA-Simple-Profile-yaml-v1.1: "TOSCA Simple Profile in YAML Version 1.1".
- [5] Private Enterprise Numbers registry at IANA.

NOTE: Available at <https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

- [6] IETF RFC 5234: "Augmented BNF for Syntax Specifications: ABNF".
- [7] ETSI GS NFV-IFA 027: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Performance Measurements Specification".
- [8] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".
- [9] IETF RFC 4122: "A Universally Unique Identifier (UUID) URN Namespace".
- [10] ISO 3166 (all parts): "Codes for the representation of names of countries and their subdivisions".
- [11] IETF RFC 4776: "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information".

NOTE: Available at <https://tools.ietf.org/html/rfc4776>.

- [12] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".
- [13] IETF RFC 5646: "Tags for Identifying Languages".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] ETSI GS NFV-IFA 007: "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".
- [i.2] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".
- [i.3] ETSI GS NFV-REL 001: "Network Functions Virtualisation (NFV); Resiliency Requirements".
- [i.4] ETSI GS NFV-SOL 002: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point".
- [i.5] Assigned Internet Protocol Numbers.

NOTE: Available at <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

- [i.6] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".
- [i.7] Mistral Workflow Language v2 specification.

NOTE: Available at https://docs.openstack.org/mistral/latest/user/wf_lang_v2.html.

- [i.8] ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

3 Definition of terms and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.2] and the following apply:

TOSCA interface type: reusable entity that describes a set of TOSCA operations that can be included as part of a Node type or Relationship Type definition

NOTE: See TOSCA-Simple-Profile-YAML-v1.2 [3].

TOSCA operation: behavioural lifecycle procedure in a TOSCA node or relationship definition that can be invoked by an orchestration engine, whose implementation definition can be provided in the service template as part of a node template definition or a relationship template definition, or rely on an implementation of the operation built in the orchestration engine

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.2] apply.

4 Overview of TOSCA model

TOSCA (Topology and Orchestration Specification for Cloud Applications) is a modelling language for describing the components of a cloud application and their relationships. TOSCA uses the concept of service templates to describe cloud workloads. TOSCA further provides means of associating standard or user-defined lifecycle operations to a cloud application component or to a relationship between components. The present document is based on TOSCA-Simple-Profile-yaml v1.2 [3], which describes a YAML rendering for TOSCA.

5 General concept of using TOSCA to model NFV descriptors

5.1 Introduction

An NFV deployment template is modelled by using one or more TOSCA service template as defined in TOSCA-Simple-Profile-yaml-v1.2 [3].

Deployment templates in NFV fully describe the attributes and requirements necessary to instantiate Network Services (NSs) and Virtualised Network Functions (VNFs) and manage their lifecycle.

Two main deployment templates are identified:

- The Virtualised Network Function Descriptor (VNFD).
- The Network Service Descriptor (NSD).

The NFVO receives these deployment templates at on-boarding time; where the VNFD is part of a VNF Package. To create a VNF instance or an NS instance, an instantiation request is sent to the VNFM or the NFVO, respectively; the request identifies the relevant deployment template and contains additional instantiation input parameters that are used to customize a specific instance of a VNF or an NS.

5.2 Network Service Descriptor

The Network Service Descriptor (NSD) is a deployment template which consists of information used by the NFVO for lifecycle management of an NS ETSI GS NFV-IFA 014 [2]. The NSD includes or references the descriptors of its constituent objects:

- Zero, one or more Virtualised Network Function Descriptors (VNFD).
- Zero, one or more Physical Network Functions Descriptors (PNFD).
- Zero, one or more nested NSD.
- Zero, one or more Virtual Link Descriptors (VLD).
- Zero, one or more VNF Forwarding Graph Descriptors (VNFFGD).

A VNFFGD describes a topology of the Network Service or a portion of the Network Service.

A VLD describes the resource for deploying and managing the lifecycle of virtual links between the constituents of an NS.

VLDs and VNFFGDs are embedded in the NSD.

A PNFD describes the connectivity requirements to integrate PNFs in an NS.

A nested NSD is an NSD from which a nested NS can be instantiated within a parent NS instance.

5.3 Virtualised Network Function Descriptor

The VNFD is a component of a VNF package. It is used by both the NFVO and the VNFM.

A VNFD is a deployment template which describes a VNF in terms of deployment and operational behaviour requirements. It also contains Virtualised Deployment Units (VDUs), internal virtual link descriptors, external connection point descriptors, software image descriptors, and deployment flavour descriptors, as defined in ETSI GS NFV-IFA 011 [1].

A VNFD contains the following main pieces of information, as shown in figure 5.3-1:

- Virtualisation Deployment Unit (VDU) is a construct supporting the description of the deployment and operational behaviour of a VNF Component (VNFC). A VNFC instance created based on the VDU maps to a single virtualisation container (e.g. a VM). A VDU describes the resources needed to deploy and manage the lifecycle of a VNFC. A VDU includes internal Connection Point Descriptors (CPDs) that describe internal connection points that can either be used to connect a VNFC to an internal virtual link or be re-exposed outside the VNF as external connection points.
- External CPD: describes an external connection point of a VNF, where either an internal connection point of a VDU is exposed as external connection point or the external connection point is directly connected to an internal virtual link.
- Internal VLD: describes the resource requirements for deploying and managing the lifecycle of virtual links between one or more VNFC instances created based on one or more VDUs.

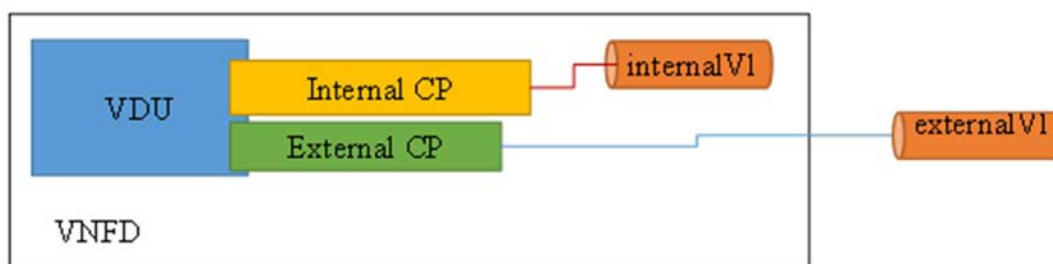


Figure 5.3-1: Overview of VNF descriptor

The information within a VNFD is structured according to one or more VNF deployment flavours (VnfDf) that specify different deployment configuration of a VNF, in terms of its internal topology and resource needs.

5.4 `tosca_definitions_version` and Namespace prefix

The "`tosca_definitions_version`" keyword when used in the present document shall comply with the definition as specified in section 3.1.2 of TOSCA-Simple-Profile-YAML-v1.2 [3] with the associated Namespace Alias value defined in the TOSCA-Simple-Profile-YAML-v1.2 specification [3].

Table 5.4-1 defines the TOSCA Namespace prefix that shall be used to declare the namespace of all the TOSCA types as specified in the present document.

Table 5.4-1

Namespace Prefix	Specification Description
<code>toscanfv</code>	The TOSCA namespace prefix of all the TOSCA types as specified in the present document.

5.5 Imports statement

5.5.1 VNFD TOSCA service template

A VNFD TOSCA service template shall include a TOSCA import definition referencing the following files:

- The file defined in clause B.2 that includes all the type definitions from clause 6 of the present document.
- Optionally, additional VNF-specific files containing type definitions used by the VNFD TOSCA service template.

As specified in [3], the import statement can use a single-line or multi-line grammar.

The single-line grammar supports the import of single or multiple uniquely named VNFD types and other template definition files.

EXAMPLE 1:

```
imports:
- https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_vnfd_2_5_1_types.yaml
- any_other_files.yaml
- custom_vnfd_datatypes_extension.yaml
```

The multi-line grammar also supports the import of single or multiple uniquely named VNFD types and other template definition files. The "file" keyword is a mandatory parameter in this grammar.

EXAMPLE 2:

```
imports:
- file: https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_vnfd_2_5_1_types.yaml
- file: any_other_files.yaml
- file: custom_vnfd_datatypes_extension.yaml
```

5.5.2 NSD TOSCA service template

A NSD TOSCA service template shall include a TOSCA import definition referencing the following files:

- The file defined in clause B.3 that includes all the type definitions from clause 7 of the present document.
- Optionally, additional NSD-specific files containing type definitions used by the NSD TOSCA service template.

As specified in [3], the import statement can use a single-line or multi-line grammar.

The single-line grammar supports the import of single or multiple uniquely named NSD types and other template definition files.

EXAMPLE 1:

```
imports:
- https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_nsd_2_5_1_types.yaml
- any_other_files.yaml
- custom_nsd_node_types_extension.yaml
```

The multi-line grammar also supports the import of single or multiple uniquely named NSD types and other template definition files. The "file" keyword is a mandatory parameter in this grammar.

EXAMPLE 2:

```
imports:
- file: https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_nsd_2_5_1_types.yaml
- file: any_other_files.yaml
- file: custom_nsd_node_types_extension.yaml
```

5.5.3 PNFD TOSCA service template

A PNFD TOSCA service template shall include a TOSCA import definition referencing the following files:

- The file defined in clause B.4 that includes all the type definitions from clause 8 of the present document.
- Optionally, additional PNFD-specific files containing type definitions used by the PNFD TOSCA service template.

As specified in [3], the import statement can use a single-line or multi-line grammar.

The single-line grammar supports the import of single or multiple uniquely named PNFD types and other template definition files.

EXAMPLE: 1

```
imports:
- https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_pnfd_2_5_1_types.yaml
- any_other_files.yaml
- custom_pnfd_node_types_extension.yaml
```

The multi-line grammar also supports the import of single or multiple uniquely named PNFD types and other template definition files. The "file" keyword is a mandatory parameter in this grammar.

EXAMPLE 2:

```
imports:
- file: https://www.etsi.org/deliver/etsi_gs/NFV-
SOL/001_099/001/02.05.01_60/etsi_nfv_sol001_pnfd_2_5_1_types.yaml
- file: any_other_files.yaml
- file: custom_pnfd_node_types_extension.yaml
```

5.6 Type extension

5.6.1 Introduction

Type extension is used when VNF-specific type information is introduced in the VNFD (e.g. modifiable attributes, configurable properties and additional parameters to LCM operations) or NSD.

5.6.2 Rules

Type extension may be applied to NFV types defined in the present document within the limits specified table 5.6.3-1, adhering to the following rule.

A derived type shall extend the base type in such a way that it remains substitutable for the base type with the following requirements.

New properties may be introduced with no restriction within the limits specified table 5.6.3-1.

Existing properties may be extended according to the following rules:

- a) A scalar property may not be extended to another type (e.g. a string property may not be replaced with an integer property or with a complex property of the same name).
- b) A complex property of data type "X" may only be extended to a property of type "Y" where "Y" is derived from "X" according to the present rules (recursive rule: present rules applied to each property of the derived data type).
- c) A property of type list with entry schema "X" may only be extended to a list with entry schema "Y" where "Y" is an extension of "X" according to the present rules (recursive rule: present rules applied to the elements of the list).
- d) A property of type map with entry schema "X" may only be extended to a map with entry schema "Y" where "Y" is an extension of "X" according to the present rules (recursive rule: present rules applied to the values of the map).

In general, the same rules (1 and 2) apply to introducing/extending other elements beyond properties such as capabilities, requirements, interfaces, operations, inputs, etc. as well; in the current version of the specification, however, there are only examples of extending properties and the `tosca.nodes.nfv.VNF` node type.

5.6.3 VNFD Types

Table 5.6.3-1 specifies the extension point where VNFD author may extend the pre-defined types.

Table 5.6.3-1: VNFD type extension points

Type	Keyname	Property name
tosca.nodes.nfv.VNF	properties	modifiable_attributes (as a new property) configurable_properties (as a new property)
	interfaces	Vnflcm.{operation_name}.inputs.additional_parameters (as a new property) Vnflcm.{operation_name}.inputs (as new properties)
tosca.nodes.nfv.Vdu.Compute	properties	configurable_properties (as a new property)
tosca.datatypes.nfv.VnfInfoModifiableAttributes	properties	extensions (as a new property) metadata (as a new property)
tosca.datatypes.nfv.VnfConfigurableProperties	properties	additional_configurable_properties (as a new property)
tosca.datatypes.nfv.VnfcConfigurableProperties	properties	additional_vnfc_configurable_properties (as a new property)
tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions	properties	(new properties)
tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata	properties	(new properties)
tosca.datatypes.nfv.VnfAdditionalConfigurableProperties	properties	(new properties)
tosca.datatypes.nfv.VnfAdditionalVnfcConfigurableProperties	properties	(new properties)
tosca.datatypes.nfv.VnfOperationAdditionalParameters	properties	(new properties)

5.6.4 NSD types

None.

6 VNFD TOSCA model

6.1 Introduction

The VNFD information model specified by ETSI GS NFV-IFA 011 [1] is mapped to the TOSCA concepts. The VNFD is represented as one or more TOSCA service templates to be used by the VNFM for deploying and managing the lifecycle of a VNF instance.

Table 6.1-1 describes the mapping of the main information elements defined in ETSI GS NFV-IFA 011 [1] applicable to a VNFD and the corresponding NFV-specific TOSCA Types, as well the basic TOSCA types defined in TOSCA Simple YAML Profile [3] from which they are derived from. The full definition of all types can be found in the following clauses.

NOTE: There is only part of VNFD features as specified in ETSI GS NFV-IFA 011 [1] is included in this version of the present document, the following are features not included:

- VnfIndicators
- autoScale

Table 6.1-1: Mapping of ETSI GS NFV-IFA 011 [1] information elements with TOSCA types

ETSI GS NFV-IFA 011 [1] Elements	VNFD TOSCA types	Derived from
VNFD	tosca.nodes.nfv.VNF	tosca.nodes.Root
Vdu	n/a (see note 1)	n/a
Cpd (Connection Point)	tosca.nodes.nfv.Cp	tosca.nodes.Root
VduCpd (internal connection point)	tosca.nodes.nfv.VduCp	tosca.nodes.nfv.Cp
VnfVirtualLinkDesc (Virtual Link)	tosca.nodes.nfv.VnfVirtualLink	tosca.nodes.Root
VnfExtCpd (External Connection Point)	tosca.nodes.nfv.VnfExtCp tosca.nodes.nfv.VduCp	tosca.nodes.nfv.Cp
VirtualStorageDesc	tosca.nodes.nfv.Vdu.VirtualBlockStorage tosca.nodes.nfv.Vdu.VirtualObjectStorage tosca.nodes.nfv.Vdu.VirtualFileStorage	tosca.nodes.Root
VirtualComputeDesc	tosca.nodes.nfv.Vdu.Compute	tosca.nodes.Root
SwImageDesc	tosca.artifacts.nfv.SwImage	tosca.artifacts.Deployment .Image
VnfDf	n/a (see note 2)	n/a
SecurityGroupRule	tosca.policies.nfv.SecurityGroupRule	tosca.policies.Root
VnfConfigurableProperties	tosca.datatypes.nfv.VnfConfigurableProperties	tosca.datatypes.Root
VnfInfoModifiableAttributes	tosca.datatypes.nfv.VnfInfoModifiableAttributes	tosca.datatypes.Root
NOTE 1: The Vdu information element is represented as a collection of toasca.nodes.nfv.VduCp, toasca.nodes.nfv.Vdu.Compute, toasca.nodes.nfv.Vdu.VirtualBlockStorage, toasca.nodes.nfv.Vdu.VirtualObjectStorage and toasca.nodes.nfv.Vdu.VirtualFileStorage types.		
NOTE 2: The VnfDf information element is represented as a TOSCA service template.		

Figure 6.1-1 provides an overview of the TOSCA node types used to build a service template representing a VNFD for a specific deployment flavour, and of the relationship between them. The figure shows one of the three types of virtual storage. A detailed description is provided in clause 6.11.

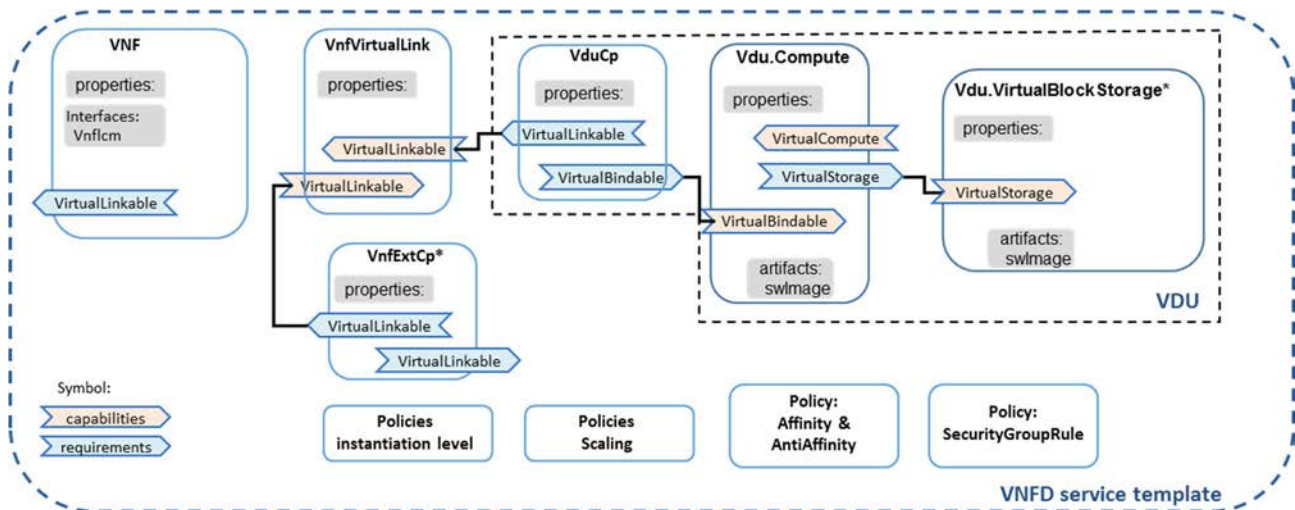


Figure 6.1-1: Service template VNFD overview

6.2 Data Types

6.2.1 toasca.datatypes.nfv.L2AddressData

6.2.1.1 Description

The L2AddressData data type describes the information on the MAC addresses to be assigned to a connection point as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.1.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.1.1-1: Type name, shorthand, and URI

Shorthand Name	L2AddressData
Type Qualified Name	toscanfv:L2AddressData
Type URI	tosca.datatypes.nfv.L2AddressData

6.2.1.2 Properties

The properties of the L2AddressData data type shall comply with the provisions set out in table 6.2.1.2-1.

Table 6.2.1.2-1: Properties

Name	Required	Type	Constraints	Description
mac_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.

6.2.1.3 Definition

The syntax of the L2AddressData data type shall comply with the following definition:

```
tosca.datatypes.nfv.L2AddressData:
  derived_from: toasca.datatypes.Root
  description: Describes the information on the MAC addresses to be assigned to a
connection point.
  properties:
```



```

mac_address_assignment:
  type: boolean
  description: Specifies 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
  required: true

```

6.2.1.4 Examples

```

<some_tosca_entity>:
  properties:
    l2_address_data:
      mac_address_assignment: true

```

6.2.1.5 Additional Requirements

None.

6.2.2 toska.datatypes.nfv.L3AddressData

6.2.2.1 Description

The L3AddressData data type supports providing information about Layer 3 level addressing scheme and parameters applicable to a CP, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.2.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.2.1-1: Type name, shorthand, and URI

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

6.2.2.2 Properties

The properties of the L3AddressData data type shall comply with the provisions set out in table 6.2.2.2-1.

Table 6.2.2.2-1: 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_protocols properties of the parent VnfExtCp.
number_of_ip_address	no	Integer	greater_than: 0	Minimum number of IP addresses to be assigned.

6.2.2.3 Definition

The syntax of the L3AddressData data type shall comply with the following definition:

```

tosca.datatypes.nfv.L3AddressData:
  derived_from: toska.datatypes.Root
  description: Provides information about Layer 3 level addressing scheme and
parameters applicable to a CP
  properties:
    ip_address_assignment:
      type: boolean
      description: Specifies 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
      required: true
    floating_ip_activated:
      type: boolean
      description: Specifies if the floating IP scheme is activated on the
Connection Point or not
      required: true
    ip_address_type:
      type: string
      description: Defines address type. The address type should be aligned with the
address type supported by the layer_protocols properties of the parent VnfExtCp
      required: false
      constraints:
        - valid_values: [ ipv4, ipv6 ]
    number_of_ip_address:
      type: integer
      description: Minimum number of IP addresses to be assigned
      required: false
      constraints:
        - greater_than: 0

```

6.2.2.4 Examples

```

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

```

6.2.2.5 Additional Requirements

None.

6.2.3 toska.datatypes.nfv.AddressData

6.2.3.1 Description

The AddressData data type describes information about the addressing scheme and parameters applicable to a CP, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.3.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.3.1-1: Type name, shorthand, and URI

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

6.2.3.2 Properties

The properties of the AddressData data type shall comply with the provisions set out in table 6.2.3.2-1.

Table 6.2.3.2-1: 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 a connection point The content type shall be aligned with the address type supported by the layerProtocol property of the connection point.
l2_address_data	no	tosca.datatypes.nfv.L2AddressData	Shall be present when the address_type is mac_address.	Provides the information on the MAC addresses to be assigned to a connection point.
l3_address_data	no	tosca.datatypes.nfv.L3AddressData	Shall be present when the address_type is ip_address.	Provides the information on the IP addresses to be assigned to a connection point.

6.2.3.3 Definition

The syntax of the AddressData data type shall comply with the following definition:

```

tosca.datatypes.nfv.AddressData:
  derived_from: toska.datatypes.Root
  description: Describes information about the addressing scheme and parameters
  applicable to a CP
  properties:
    address_type:
      type: string
      description: Describes the type of the address to be assigned to a connection
      point. The content type shall be aligned with the address type supported by the
      layerProtocol property of the connection point
      required: true
      constraints:
        - valid_values: [ mac_address, ip_address ]
    l2_address_data:
      type: toska.datatypes.nfv.L2AddressData
      description: Provides the information on the MAC addresses to be assigned to a
      connection point.
      required: false
    l3_address_data:
      type: toska.datatypes.nfv.L3AddressData
      description: Provides the information on the IP addresses to be assigned to a
      connection point
      required: false

```

6.2.3.4 Examples

```

<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
    number_of_ip_address: 4

```

6.2.3.5 Additional Requirements

None.

6.2.4 tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements

6.2.4.1 Description

The VirtualNetworkInterfaceRequirements data type describes requirements on a virtual network interface, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.4.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.4.1-1: Type name, shorthand, and URI

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

6.2.4.2 Properties

The properties of the VirtualNetworkInterfaceRequirements data type shall comply with the provisions set out in table 6.2.4.2-1.

Table 6.2.4.2-1: 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		Indicates whether fulfilling the constraint is mandatory (TRUE) for successful operation or desirable (FALSE).
network_interface_requirements	yes	map of string		The network interface requirements. A map of strings that contain a set of key-value pairs that describes the hardware platform specific network interface deployment requirements. More information regarding the usage of this property is available at: https://register.etsi.org
nic_io_requirements	no	tosca.datatypes.nfv.LogicalNodeData		This references (couples) the CP with any logical node I/O requirements (for network devices) that may have been created. Linking these attributes is necessary so that I/O requirements that need to be articulated at the logical node level can be associated with the network interface requirements associated with the CP.

6.2.4.3 Definition

The syntax of the VirtualNetworkInterfaceRequirements data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements:
  derived_from: toska.datatypes.Root
  description: Describes requirements on a virtual network interface
  properties:
    name:
      type: string
      description: Provides a human readable name for the requirement.
      required: false
    description:
      type: string
      description: Provides a human readable description of the requirement.
      required: false
    support_mandatory:
      type: boolean
      description: Indicates whether fulfilling the constraint is mandatory (TRUE)
for successful operation or desirable (FALSE).
      required: true
    network_interface_requirements:
      type: map
      description: The network interface requirements. A map of strings that contain
a set of key-value pairs that describes the hardware platform specific network
interface deployment requirements.
      required: true
      entry_schema:
        type: string
    nic_io_requirements:
      type: toska.datatypes.nfv.LogicalNodeData
      description: references (couples) the CP with any logical node I/O
requirements (for network devices) that may have been created. Linking these
attributes is necessary so that so that I/O requirements that need to be articulated
at the logical node level can be associated with the network interface requirements
associated with the CP.
      required: false

```

6.2.4.4 Examples

None.

6.2.4.5 Additional Requirements

None.

6.2.5 toska.datatypes.nfv.ConnectivityType

6.2.5.1 Description

The ConnectivityType data type describes additional connectivity information of a virtualLink, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.5.1-1: Type name, shorthand, and URI

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

6.2.5.2 Properties

The properties of the ConnectivityType shall comply with the provisions set out in table 6.2.5.2-1.

Table 6.2.5.2-1: Properties

Name	Required	Type	Constraints	Description
layer_protocols	yes	list of string	Valid values: ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire	Identifies the protocol a virtualLink gives access to (ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire). The top layer protocol of the virtualLink protocol stack shall always be provided. The lower layer protocols may be included when there are specific requirements on these layers.
flow_pattern	no	string	Valid values: line, tree, mesh	Identifies the flow pattern of the connectivity.

6.2.5.3 Definition

The syntax of the ConnectivityType data type shall comply with the following definition:

```

tosca.datatypes.nfv.ConnectivityType:
  derived_from: toasca.datatypes.Root
  description: describes additional connectivity information of a virtualLink
  properties:
    layer_protocols:
      type: list
      description: Identifies the protocol a virtualLink gives access to (ethernet,
mpls, odu2, ipv4, ipv6, pseudo-wire). The top layer protocol of the virtualLink
protocol stack shall always be provided. The lower layer protocols may be included
when there are specific requirements on these layers.
      required: true
      entry_schema:
        type: string
      constraints:
        - valid_values: [ ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire ]
    flow_pattern:
      type: string
      description: Identifies the flow pattern of the connectivity
      required: false
      constraints:
        - valid_values: [ line, tree, mesh ]

```

6.2.5.4 Examples

```

<some_tosca_entity>:
  properties:
    connectivity_type:
      layer_protocol:
        - ipv4
      flow_pattern: mesh

```

6.2.5.5 Additional Requirements

None.

6.2.6 toska.datatypes.nfv.RequestedAdditionalCapability

6.2.6.1 Description

The RequestedAdditionalCapability data type describes requested additional capability for a particular VDU, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.6.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.6.1-1: Type name, shorthand, and URI

Shorthand Name	RequestedAdditionalCapability
Type Qualified Name	toscanfv:RequestedAdditionalCapability
Type URI	tosca.datatypes.nfv.RequestedAdditionalCapability

6.2.6.2 Properties

The properties of the RequestedAdditionalCapability data type shall comply with the provisions set out in table 6.2.6.2-1.

Table 6.2.6.2-1: Properties

Name	Required	Type	Constraints	Description
requested_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_additional_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.

6.2.6.3 Definition

The syntax of the RequestedAdditionalCapability data type shall comply with the following definition:

```
tosca.datatypes.nfv.RequestedAdditionalCapability:
  derived_from: toska.datatypes.Root
  description: describes requested additional capability for a particular VDU
  properties:
    requested_additional_capability_name:
      type: string
      description: Identifies a requested additional capability for the VDU.
      required: true
    support_mandatory:
      type: boolean
      description: Indicates whether the requested additional capability is
mandatory for successful operation.
      required: true
    min_requested_additional_capability_version:
      type: string
      description: Identifies the minimum version of the requested additional
capability.
      required: false
    preferred_requested_additional_capability_version:
      type: string
      description: Identifies the preferred version of the requested additional
capability.
```

```

    required: false
    target_performance_parameters:
      type: map
      description: Identifies specific attributes, dependent on the requested
additional capability type.
      required: true
    entry_schema:
      type: string

```

6.2.6.4 Examples

None.

6.2.6.5 Additional Requirements

None.

6.2.7 `tosca.datatypes.nfv.VirtualMemory`

6.2.7.1 Description

The VirtualMemory data typesupports the specification of requirements related to virtual memory of a virtual compute resource, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.7.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.7.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualMemory
Type Qualified Name	toscanfv:VirtualMemory
Type URI	tosca.datatypes.nfv.VirtualMemory

6.2.7.2 Properties

The properties of the VirtualMemory data type shall comply with the provisions set out in table 6.2.7.2-1.

Table 6.2.7.2-1: Properties

Name	Required	Type	Constraints	Description
virtual_mem_size	yes	scalar-unit.size		Amount of virtual memory.
virtual_mem_oversubscription_policy	no	string		The memory core oversubscription policy in terms of virtual memory to physical memory on the platform.
vdu_mem_requirements	no	map of string		The hardware platform specific VDU memory requirements. A map of strings that contains a set of key-value pairs that describes hardware platform specific VDU memory requirements. More information regarding the usage of this property is available at: https://register.etsi.org
numa_enabled	no	boolean	default: false	It specifies the memory allocation to be cognisant of the relevant process/core allocation.

6.2.7.3 Definitions

The syntax of the VirtualMemory data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualMemory:
  derived_from: tosca.datatypes.Root
  description: supports the specification of requirements related to virtual
memory of a virtual compute resource
  properties:
    virtual_mem_size:
      type: scalar-unit.size
      description: Amount of virtual memory.
      required: true
    virtual_mem_oversubscription_policy:
      type: string
      description: The memory core oversubscription policy in terms of virtual
memory to physical memory on the platform.
      required: false
    vdu_mem_requirements:
      type: map
      description: The hardware platform specific VDU memory requirements. A map
of strings that contains a set of key-value pairs that describes hardware platform
specific VDU memory requirements.
      required: false
      entry_schema:
        type: string
    numa_enabled:
      type: boolean
      description: It specifies the memory allocation to be cognisant of the
relevant process/core allocation.
      required: false
      default: false

```

6.2.7.4 Examples

None.

6.2.7.5 Additional Requirements

None.

6.2.8 tosca.datatypes.nfv.VirtualCpu

6.2.8.1 Description

The VirtualCpu data type supports the specification of requirements related to virtual CPU(s) of a virtual compute resource, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.8.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.8.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualCpu
Type Qualified Name	toscanfv:VirtualCpu
Type URI	tosca.datatypes.nfv.VirtualCpu

6.2.8.2 Properties

The properties of the VirtualCpu data type shall comply with the provisions set out in table 6.2.8.2-1.

Table 6.2.8.2-1: Properties

Name	Required	Type	Constraints	Description
cpu_architecture	no	string		CPU architecture type. Examples are x86, ARM.
num_virtual_cpu	yes	integer	greater_than: 0	Number of virtual CPUs.
virtual_cpu_clock	no	scalar-unit.frequency		Minimum virtual CPU clock rate.
virtual_cpu_oversubscription_policy	no	string		CPU core oversubscription policy e.g. the relation of virtual CPU cores to physical CPU cores/threads.
vdu_cpu_requirements	no	map of string		The hardware platform specific VDU CPU requirements. A map of strings that contains a set of key-value pairs describing VDU CPU specific hardware platform requirements. More information regarding the usage of this property is available at: https://register.etsi.org
virtual_cpu_pinning	no	tosca.datatypes.nfv.VirtualCpuPinning		The virtual CPU pinning configuration for the virtualised compute resource.

6.2.8.3 Definition

The syntax of the VirtualCpu data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualCpu:
  derived_from: tosc.datatypes.Root
  description: Supports the specification of requirements related to virtual CPU(s)
of a virtual compute resource
  properties:
    cpu_architecture:
      type: string
      description: CPU architecture type. Examples are x86, ARM
      required: false
    num_virtual_cpu:
      type: integer
      description: Number of virtual CPUs
      required: true
      constraints:
        - greater_than: 0
    virtual_cpu_clock:
      type: scalar-unit.frequency
      description: Minimum virtual CPU clock rate
      required: false
    virtual_cpu_oversubscription_policy:
      type: string
      description: CPU core oversubscription policy e.g. the relation of virtual CPU
cores to physical CPU cores/threads.
      required: false
    vdu_cpu_requirements:
      type: map
      description: The hardware platform specific VDU CPU requirements. A map of
strings that contains a set of key-value pairs describing VDU CPU specific hardware
platform requirements.
      required: false
    entry_schema:

```

```

    type: string
  virtual_cpu_pinning:
    type: toasca.datatypes.nfv.VirtualCpuPinning
    description: The virtual CPU pinning configuration for the virtualised compute
resource.
    required: false

```

6.2.8.4 Examples

None.

6.2.8.5 Additional Requirements

None.

6.2.9 toasca.datatypes.nfv.VirtualCpuPinning

6.2.9.1 Description

The VirtualCpuPinning data type supports the specification of requirements related to the virtual CPU pinning configuration of a virtual compute resource, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.9.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.9.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualCpuPinning
Type Qualified Name	toscanfv:VirtualCpuPinning
Type URI	tosca.datatypes.nfv.VirtualCpuPinning

6.2.9.2 Properties

The properties of the VirtualCpuPinning data type shall comply with the provisions set out in table 6.2.9.2-1.

Table 6.2.9.2-1: Properties

Name	Required	Type	Constraints	Description
virtual_cpu_pinning_policy	no	string	Valid values: static, dynamic	Indicates the policy for CPU pinning. The policy can take values of "static" or "dynamic". In case of "dynamic" the allocation of virtual CPU cores to logical CPU cores is decided by the VIM. (e.g. SMT (Simultaneous Multi-Threading) requirements). In case of "static" the allocation is requested to be according to the <code>virtual_cpu_pinning_rule</code> .
virtual_cpu_pinning_rule	no	list of string		Provides the list of rules for allocating virtual CPU cores to logical CPU cores/threads.

6.2.9.3 Definition

The syntax of the VirtualCpuPinning data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualCpuPinning:
  derived_from: toasca.datatypes.Root
  description: Supports the specification of requirements related to the virtual CPU
pinning configuration of a virtual compute resource
  properties:
    virtual_cpu_pinning_policy:

```

```

    type: string
    description: Indicates the policy for CPU pinning. The policy can take values
of "static" or "dynamic". In case of "dynamic" the allocation of virtual CPU cores to
logical CPU cores is decided by the VIM. (e.g. SMT (Simultaneous Multi-Threading)
requirements). In case of "static" the allocation is requested to be according to the
virtual_cpu_pinning_rule.
    required: false
    constraints:
      - valid_values: [ static, dynamic ]
  virtual_cpu_pinning_rule:
    type: list
    description: Provides the list of rules for allocating virtual CPU cores to
logical CPU cores/threads
    required: false
    entry_schema:
      type: string

```

6.2.9.4 Examples

None.

6.2.9.5 Additional Requirements

The `virtual_cpu_pinning_rule` shall be included if the `virtual_cpu_pinning_policy` property is set to "static" and shall be absent otherwise.

6.2.10 `tosca.datatypes.nfv.VnfcConfigurableProperties`

6.2.10.1 Description

The `VnfcConfigurableProperties` data type defines the configurable properties of a VNFC, as defined in ETSI GS NFV-IFA 011 [1].

Table 6.2.10.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.10.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfcConfigurableProperties</code>
Type Qualified Name	<code>toscanfv:VnfcConfigurableProperties</code>
Type URI	<code>tosca.datatypes.nfv.VnfcConfigurableProperties</code>

6.2.10.2 Properties

The properties of the `VnfcconfigurableProperties` shall comply with the provisions set out in table 6.2.10.2-1.

Table 6.2.10.2-1: Properties

Name	Required	Type	Constraints	Description
<code>additional_vnfc_configurable_properties</code>	no	<code>tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties</code>		Describes additional configuration for VNFC that can be modified using the <code>ModifyVnfcInfo</code> operation.

6.2.10.3 Definition

The syntax of the VnfcConfigurableProperties data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfcConfigurableProperties:
  derived_from: tosca.datatypes.Root
  description: Defines the configurable properties of a VNFC
  #properties:
    # additional_vnfc_configurable_properties:
    #   type: tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties
    #   description: Describes additional configuration for VNFC that can be
modified using the ModifyVnfInfo operation
    #   required: false
    #   derived types are expected to introduce
    #   additional_vnfc_configurable_properties with its type derived from
    #   tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties

```

6.2.10.4 Examples

```

tosca_definitions_version: tosca_simple_yaml_1_2

node_types:
  MyCompany.nodes.nfv.Vdu.Aux:
    derived_from: tosca.nodes.nfv.Vdu.Compute
    properties:
      configurable_properties:
        type: MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties

data_types:
  MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfcConfigurableProperties
    properties:
      additional_vnfc_configurable_properties:
        type: MyCompany.datatypes.nfv.AuxVnfcAdditionalConfigurableProperties
        required: true

  MyCompany.datatypes.nfv.AuxVnfcAdditionalConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties
    properties:
      name_prefix_in_vim:
        type: string
        required: false
      dns_server:
        type: string
        required: true

topology_template:
  ..

node_templates:
  aux:
    type: MyCompany.nodes.nfv.Vdu.Aux
    properties:
      ..

```

6.2.10.5 Additional Requirements

None.

6.2.11 `tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties`

6.2.11.1 Description

The `VnfcAdditionalConfigurableProperties` type is an empty base type for deriving data types for describing additional configurable properties for a given VNFC. Table 6.2.11.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.11.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfcAdditionalConfigurableProperties</code>
Type Qualified Name	<code>toscanfv:VnfcAdditionalConfigurableProperties</code>
Type URI	<code>tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties</code>

6.2.11.2 Properties

None.

6.2.11.3 Definition

The syntax of the `VnfcAdditionalConfigurableProperties` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties:
  derived_from: toska.datatypes.Root
  description: VnfcAdditionalConfigurableProperties type is an empty base type
for deriving data types for describing additional configurable properties for a
given VNFC.
```

6.2.12 `tosca.datatypes.nfv.VduProfile`

6.2.12.1 Description

The `VduProfile` data type describes additional instantiation data for a given `Vdu.Compute` used in a specific deployment flavour. Table 6.2.12.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.12.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VduProfile</code>
Type Qualified Name	<code>toscanfv:VduProfile</code>
Type URI	<code>tosca.datatypes.nfv.VduProfile</code>

6.2.12.2 Properties

The properties of the `VduProfile` data type shall comply with the provisions set out in table 6.2.12.2-1.

Table 6.2.12.2-1: Properties

Name	Required	Type	Constraints	Description
min_number_of_instances	yes	integer	greater_or_equal: 0	Minimum number of instances of the VNFC based on this Vdu.Compute that is permitted to exist for a particular VNF deployment flavour.
max_number_of_instances	yes	integer	greater_or_equal: 0	Maximum number of instances of the VNFC based on this Vdu.Compute that is permitted to exist for a particular VNF deployment flavour.
NOTE: A vduId property, which exists in ETSI GS NFV-IFA 011 [1] is not needed, as the VduProfile is contained in the Vdu.Compute node.				

6.2.12.3 Definition

The syntax of the VduProfile data type shall comply with the following definition:

```

tosca.datatypes.nfv.VduProfile:
  derived_from: tosca.datatypes.Root
  description: describes additional instantiation data for a given Vdu.Compute used
in a specific deployment flavour.
  properties:
    min_number_of_instances:
      type: integer
      description: Minimum number of instances of the VNFC based on this Vdu.Compute
that is permitted to exist for a particular VNF deployment flavour.
      required: true
      constraints:
        - greater_or_equal: 0
    max_number_of_instances:
      type: integer
      description: Maximum number of instances of the VNFC based on this Vdu.Compute
that is permitted to exist for a particular VNF deployment flavour.
      required: true
      constraints:
        - greater_or_equal: 0

```

6.2.12.4 Examples

```

tosca_definitions_version: tosca_simple_yaml_1_2

topology template:
...

node_templates:
  VDU_A:
    type: tosca.nodes.nfv.Vdu.Compute
    properties:
      vdu_profile:
        min_number_of_instances: 2
        max_number_of_instances: 6
      # other properties omitted for brevity
    requirements:
      - virtual_storage: VirtualStorage_A1
      - virtual_storage: VirtualStorage_A2
    capabilities:
      virtual_binding:

```

```

VirtualStorage_A1:
  type: toasca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    # omitted for brevity
  capabilities:
    virtual_storage

VirtualStorage_A2:
  type: toasca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    # omitted for brevity
  capabilities:
    virtual_storage

VduCp_A1:
  type: toasca.nodes.nfv.VduCp
  properties:
    # omitted for brevity
  requirements:
    - virtual_binding: VDU_A
    - virtual_link

VduCp_A2:
  type: toasca.nodes.nfv.VduCp
  properties:
    # omitted for brevity
  requirements:
    - virtual_binding: VDU_A
    - virtual_link

```

Above snippet shows part of a topology template. The VDU_A node template is a Vdu.Compute node that is connected to two VirtualBlockStorage nodes: VirtualStorage_A1 and VirtualStorage_A2. It also has two VduCps: VduCp_A1 and Vdu_CpA2.

The minimum number of instances of VDU_A that are permitted to exist is 2. Likewise, the minimum number of instances of VirtualStorage_A1, VirtualStorage_A2, VduCp_A1 and VduCp_A2 that are permitted to exist is 2.

The maximum number of instances of VDU_A that are permitted to exist is 6. Likewise, the maximum number of instances of VirtualStorage_A1, VirtualStorage_A2, VduCp_A1 and VduCp_A2 that are permitted to exist is 6.

6.2.12.5 Additional requirements

The properties of the vdu_profile indicate the maximum and minimum number of Vdu.Compute instances that are permitted to exist, created from a given Vdu.Compute node template during its lifecycle, as well as:

- The maximum and minimum number of instances of each VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes connected to the Vdu.Compute via one particular occurrence of the virtual_storage requirement.
- The maximum and minimum number instances of each VduCp node connected to the Vdu.Compute via one particular occurrence of the virtual_binding capability.

6.2.13 tosca.datatypes.nfv.VIProfile

6.2.13.1 Description

The VIProfile data type describes additional instantiation data for a given VL used in a specific VNF deployment flavour. Table 6.2.13.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.13.1-1: Type name, shorthand, and URI

Shorthand Name	VIProfile
Type Qualified Name	toscanfv:VIProfile
Type URI	tosca.datatypes.nfv.VIProfile

6.2.13.2 Properties

The properties of the VIProfile data type shall comply with the provisions set out in table 6.2.13.2-1.

Table 6.2.13.2-1: Properties

Name	Required	Type	Constraints	Description
max_bitrate_requirements	yes	tosca.datatypes.nfv.LinkBitrateRequirements		Specifies the maximum bitrate requirements for a VL instantiated according to this profile.
min_bitrate_requirements	yes	tosca.datatypes.nfv.LinkBitrateRequirements		Specifies the minimum bitrate requirements for a VL instantiated according to this profile.
qos	no	tosca.datatypes.nfv.Qos		Specifies the QoS requirements of a VL instantiated according to this profile.
virtual_link_protocol_data	no	list of tosca.datatypes.nfv.VirtualLinkProtocolData		Specifies the protocol data for a virtual link.
NOTE: A vnfVirtualLinkDescId property, which exists in ETSI GS NFV-IFA 011 [1] is not needed, as the VLProfile is contained in the VL node.				

6.2.13.3 Definition

The syntax of the VIProfile data type shall comply with the following definition:

```
tosca.datatypes.nfv.VIProfile:
  derived_from: tosca.datatypes.Root
  description: Describes additional instantiation data for a given VL used in a
  specific VNF deployment flavour.
  properties:
    max_bitrate_requirements:
      type: tosca.datatypes.nfv.LinkBitrateRequirements
      description: Specifies the maximum bitrate requirements for a VL instantiated
  according to this profile.
      required: true
    min_bitrate_requirements:
      type: tosca.datatypes.nfv.LinkBitrateRequirements
      description: Specifies the minimum bitrate requirements for a VL instantiated
  according to this profile.
      required: true
    qos:
      type: tosca.datatypes.nfv.Qos
      description: Specifies the QoS requirements of a VL instantiated according to
  this profile.
      required: false
```

```

virtual_link_protocol_data:
  type: list
  description: Specifies the protocol data for a virtual link.
  required: false
  entry_schema:
    type: tosca.datatypes.nfv.VirtualLinkProtocolData

```

6.2.13.4 Examples

None.

6.2.13.5 Additional Requirements

None.

6.2.14 tosca.datatypes.nfv.VirtualLinkProtocolData

6.2.14.1 Description

The VirtualLinkProtocolData data type describes one protocol layer and associated protocol data for a given virtual link used in a specific VNF deployment flavour. Table 6.2.14.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.14.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualLinkProtocolData
Type Qualified Name	toscanfv:VirtualLinkProtocolData
Type URI	tosca.datatypes.nfv.VirtualLinkProtocolData

6.2.14.2 Properties

The properties of the VirtualLinkProtocolData data type shall comply with the provisions set out in table 6.2.14.2-1.

Table 6.2.14.2-1: Properties

Name	Required	Type	Constraints	Description
associated_layer_protocol	yes	string	Valid values: ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire	Identifies one of the protocols a virtualLink gives access to (ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire) as specified by the connectivity_type property.
l2_protocol_data	no	tosca.datatypes.nfv.L2ProtocolData		Specifies the L2 protocol data for a virtual link. Shall be present when the associatedLayerProtocol attribute indicates a L2 protocol and shall be absent otherwise.
l3_protocol_data	no	tosca.datatypes.nfv.L3ProtocolData		Specifies the L3 protocol data for this virtual link. Shall be present when the associatedLayerProtocol attribute indicates a L3 protocol and shall be absent otherwise.

6.2.14.3 Definition

The syntax of the VirtualLinkProtocolData data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualLinkProtocolData:
  derived_from: tosca.datatypes.Root
  description: describes one protocol layer and associated protocol data for a
  given virtual link used in a specific VNF deployment flavour

```

```

properties:
  associated_layer_protocol:
    type: string
    description: Identifies one of the protocols a virtualLink gives access
to (ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire) as specified by the
connectivity_type property.
    required: true
    constraints:
      - valid_values: [ ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire ]
  l2_protocol_data:
    type: toasca.datatypes.nfv.L2ProtocolData
    description: Specifies the L2 protocol data for a virtual link. Shall be
present when the associatedLayerProtocol attribute indicates a L2 protocol and
shall be absent otherwise.
    required: false
  l3_protocol_data:
    type: toasca.datatypes.nfv.L3ProtocolData
    description: Specifies the L3 protocol data for this virtual link. Shall
be present when the associatedLayerProtocol attribute indicates a L3 protocol and
shall be absent otherwise.
    required: false

```

6.2.14.4 Examples

None.

6.2.14.5 Additional Requirements

None.

6.2.15 toasca.datatypes.nfv.L2ProtocolData

6.2.15.1 Description

The L2ProtocolData data type describes L2 protocol data for a given virtual link used in a specific VNF deployment flavour. Table 6.2.15.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.15.1-1: Type name, shorthand, and URI

Shorthand Name	L2ProtocolData
Type Qualified Name	toscanfv:L2ProtocolData
Type URI	tosca.datatypes.nfv.L2ProtocolData

6.2.15.2 Properties

The properties of the L2ProtocolData data type shall comply with the provisions set out in table 6.2.15.2-1.

Table 6.2.15.2-1: Properties

Name	Required	Type	Constraints	Description
name	no	string		Identifies the network name associated with this L2 protocol.
network_type	no	string	Valid values: flat, vlan, vxlan, gre	Specifies the network type for this L2 protocol. The value may be overridden at run-time.
vlan_transparent	no	boolean	default: false	Specifies whether to support VLAN transparency for this L2 protocol or not.
mtu	no	integer	greater_than : 0	Specifies the maximum transmission unit (MTU) value for this L2 protocol.

6.2.15.3 Definition

The syntax of the L2ProtocolData data type shall comply with the following definition:

```

tosca.datatypes.nfv.L2ProtocolData:
  derived_from: toska.datatypes.Root
  description: describes L2 protocol data for a given virtual link used in a
  specific VNF deployment flavour.
  properties:
    name:
      type: string
      description: Identifies the network name associated with this L2 protocol.
      required: false
    network_type:
      type: string
      description: Specifies the network type for this L2 protocol. The value may
  be overridden at run-time.
      required: false
      constraints:
        - valid_values: [ flat, vlan, vxlan, gre ]
    vlan_transparent:
      type: boolean
      description: Specifies whether to support VLAN transparency for this L2
  protocol or not.
      required: false
      default: false
    mtu:
      type: integer
      description: Specifies the maximum transmission unit (MTU) value for this
  L2 protocol.
      required: false
      constraints:
        - greater_than: 0

```

6.2.15.4 Examples

None.

6.2.15.5 Additional Requirements

None.

6.2.16 tosca.datatypes.nfv.L3ProtocolData

6.2.16.1 Description

The L3ProtocolData data type describes L3 protocol data for a given virtual link used in a specific VNF deployment flavour. Table 6.2.16.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.16.1-1: Type name, shorthand, and URI

Shorthand Name	L3ProtocolData
Type Qualified Name	toscanfv:L3ProtocolData
Type URI	tosca.datatypes.nfv.L3ProtocolData

6.2.16.2 Properties

The properties of the L3ProtocolData data type shall comply with the provisions set out in table 6.2.16.2-1.

Table 6.2.16.2-1: Properties

Name	Required	Type	Constraints	Description
name	no	string		Identifies the network name associated with this L3 protocol.
ip_version	yes	string	Valid values: ipv4 ipv6	Specifies IP version of this L3 protocol. The value of the ip_version property shall be consistent with the value of the layer_protocol in the connectivity_type property of the virtual link node.
cidr	yes	string		Specifies the CIDR (Classless Inter-Domain Routing) of this L3 protocol. The value may be overridden at run-time.
ip_allocation_pools	no	list of tosca.datatypes.nfv.IpAllocationPool		Specifies the allocation pools with start and end IP addresses for this L3 protocol. The value may be overridden at run-time.
gateway_ip	no	string		Specifies the gateway IP address for this L3 protocol. The value may be overridden at run-time.
dhcp_enabled	no	boolean		Indicates whether DHCP (Dynamic Host Configuration Protocol) is enabled or disabled for this L3 protocol. The value may be overridden at run-time.
ipv6_address_mode	no	string	Valid values: slaac dhcpv6-stateful dhcpv6-stateless	Specifies IPv6 address mode. May be present when the value of the ipVersion attribute is "ipv6" and shall be absent otherwise. The value may be overridden at run-time.

6.2.16.3 Definition

The syntax of the L3ProtocolData data type shall comply with the following definition:

```
tosca.datatypes.nfv.L3ProtocolData:
  derived_from: tosca.datatypes.Root
  description: describes L3 protocol data for a given virtual link used in a
  specific VNF deployment flavour.
  properties:
    name:
      type: string
```

```

    description: Identifies the network name associated with this L3 protocol.
    required: false
  ip_version:
    type: string
    description: Specifies IP version of this L3 protocol. The value of the
ip_version property shall be consistent with the value of the layer_protocol in
the connectivity_type property of the virtual link node.
    required: true
    constraints:
      - valid_values: [ ipv4, ipv6 ]
  cidr:
    type: string
    description: Specifies the CIDR (Classless Inter-Domain Routing) of this
L3 protocol. The value may be overridden at run-time.
    required: true
  ip_allocation_pools:
    type: list
    description: Specifies the allocation pools with start and end IP
addresses for this L3 protocol. The value may be overridden at run-time.
    required: false
    entry_schema:
      type: tosca.datatypes.nfv.IpAllocationPool
  gateway_ip:
    type: string
    description: Specifies the gateway IP address for this L3 protocol. The
value may be overridden at run-time.
    required: false
  dhcp_enabled:
    type: boolean
    description: Indicates whether DHCP (Dynamic Host Configuration Protocol)
is enabled or disabled for this L3 protocol. The value may be overridden at run-
time.
    required: false
  ipv6_address_mode:
    type: string
    description: Specifies IPv6 address mode. May be present when the value of
the ipVersion attribute is "ipv6" and shall be absent otherwise. The value may be
overridden at run-time.
    required: false
    constraints:
      - valid_values: [ slaac, dhcpv6-stateful, dhcpv6-stateless ]

```

6.2.16.4 Examples

None.

6.2.16.5 Additional Requirements

None.

6.2.17 tosca.datatypes.nfv.IpAllocationPool

6.2.17.1 Description

The IpAllocationPool data type specifies a range of IP addresses. Table 6.2.17.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.17.1-1: Type name, shorthand, and URI

Shorthand Name	IpAllocationPool
Type Qualified Name	toscanfv:IpAllocationPool
Type URI	tosca.datatypes.nfv.IpAllocationPool

6.2.17.2 Properties

The properties of the IpAllocationPool data type shall comply with the provisions set out in table 6.2.17.2-1.

Table 6.2.17.2-1: Properties

Name	Required	Type	Constraints	Description
start_ip_address	yes	string		The IP address to be used as the first one in a pool of addresses derived from the cidr block full IP range
end_ip_address	yes	string		The IP address to be used as the last one in a pool of addresses derived from the cidr block full IP range

6.2.17.3 Definition

The syntax of the IpAllocationPool data type shall comply with the following definition:

```
tosca.datatypes.nfv.IpAllocationPool:
  derived_from: toscanfv.Root
  description: Specifies a range of IP addresses
  properties:
    start_ip_address:
      type: string
      description: The IP address to be used as the first one in a pool of
addresses derived from the cidr block full IP range
      required: true
    end_ip_address:
      type: string
      description: The IP address to be used as the last one in a pool of
addresses derived from the cidr block full IP range
      required: true
```

6.2.17.4 Examples

None.

6.2.17.5 Additional Requirements

None.

6.2.18 toscanfv.InstantiationLevel

6.2.18.1 Description

The InstantiationLevel data type describes the scale level for each aspect that corresponds to a given level of resources to be instantiated within a deployment flavour in term of the number VNFC instances . Table 6.2.18.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.18.1-1: Type name, shorthand, and URI

Shorthand Name	InstantiationLevel
Type Qualified Name	toscanfv:InstantiationLevel
Type URI	tosca.datatypes.nfv.InstantiationLevel

6.2.18.2 Properties

The properties of the InstantiationLevel data type shall comply with the provisions set out in table 6.2.18.2-1.

Table 6.2.18.2-1: Properties

Name	Required	Type	Constraints	Description
description	yes	string		Human readable description of the level.
scale_info	no	map of tosca.datatypes .nfv.ScaleInfo		Represents for each aspect the scale level that corresponds to this instantiation level. scale_info shall be present if the VNF supports scaling.

6.2.18.3 Definition

The syntax of the InstantiationLevel data type shall comply with the following definition:

```

tosca.datatypes.nfv.InstantiationLevel:
  derived_from: toasca.datatypes.Root
  description: Describes the scale level for each aspect that corresponds to a given
level of resources to be instantiated within a deployment flavour in term of the
number VNFC instances
  properties:
    description:
      type: string
      description: Human readable description of the level
      required: true
    scale_info:
      type: map # key: aspectId
      description: Represents for each aspect the scale level that corresponds to
this instantiation level. scale_info shall be present if the VNF supports scaling.
      required: false
    entry_schema:
      type: toasca.datatypes.nfv.ScaleInfo

```

6.2.18.4 Examples

See clause A.6.

6.2.18.5 Additional Requirements

None.

6.2.19 toasca.datatypes.nfv.VduLevel

6.2.19.1 Description

The VduLevel data type indicates for a given Vdu.Compute in a given level the number of instances to deploy. Table 6.2.19.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.19.1-1: Type name, shorthand, and URI

Shorthand Name	VduLevel
Type Qualified Name	toscanfv:VduLevel
Type URI	tosca.datatypes.nfv.VduLevel

6.2.19.2 Properties

The properties of the VduLevel data type shall comply with the provisions set out in table 6.2.19.2-1.

Table 6.2.19.2-1: Properties

Name	Required	Type	Constraints	Description
number_of_instances	yes	integer	greater_or_equal : 0	Number of instances of VNFC based on this VDU to deploy for this level.

6.2.19.3 Definition

The syntax of the VduLevel data type shall comply with the following definition:

```

tosca.datatypes.nfv.VduLevel:
  derived_from: toscanfv.Root
  description: Indicates for a given Vdu.Compute in a given level the number of
instances to deploy
  properties:
    number_of_instances:
      type: integer
      description: Number of instances of VNFC based on this VDU to deploy for this
level.
    required: true
    constraints:
      - greater_or_equal: 0

```

6.2.19.4 Examples

See clause A.6.

6.2.19.5 Additional Requirements

None.

6.2.20 toscanfv.VnfLcmOperationsConfiguration

6.2.20.1 Description

The VnfLcmOperationsConfiguration data type represents information to configure lifecycle management operations as specified in ETSI GS NFV-IFA 007 [i.1]. Each VNF LCM operations configuration property represents a container for all attributes that affect the invocation of the corresponding VNF Lifecycle Management operation. Table 6.2.20.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.20.1-1: Type name, shorthand, and URI

Shorthand Name	VnfLcmOperationsConfiguration
Type Qualified Name	toscanfv:VnfLcmOperationsConfiguration
Type URI	tosca.datatypes.nfv.VnfLcmOperationsConfiguration

6.2.20.2 Properties

The properties of the VnfLcmOperationsConfiguration data type shall comply with the provisions set out in table 6.2.20.2-1.

Table 6.2.20.2-1: Properties

Name	Required	Type	Constraints	Description
instantiate	no	tosca.datatypes.nfv.VnfInstantiateOperationConfiguration		Configuration parameters for the InstantiateVnf operation.
scale	no	tosca.datatypes.nfv.VnfScaleOperationConfiguration		Configuration parameters for the ScaleVnf operation.
scale_to_level	no	tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration		Configuration parameters for the ScaleVnfToLevel operation.
change_flavour	no	tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration		Configuration parameters for the changeVnfFlavourOpConfig operation.
heal	no	tosca.datatypes.nfv.VnfHealOperationConfiguration		Configuration parameters for the HealVnf operation.
terminate	no	tosca.datatypes.nfv.VnfTerminateOperationConfiguration		Configuration parameters for the TerminateVnf operation.
operate	no	tosca.datatypes.nfv.VnfOperateOperationConfiguration		Configuration parameters for the OperateVnf operation.
change_ext_connectivity	no	tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration		Configuration parameters for the changeExtVnfConnectivityOpConfig operation.

6.2.20.3 Definition

The syntax of the VnfLcmOperationsConfiguration data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfLcmOperationsConfiguration:
  derived_from: toska.datatypes.Root
  description: Represents information to configure lifecycle management operations
  properties:
    instantiate:
      type: toska.datatypes.nfv.VnfInstantiateOperationConfiguration
      description: Configuration parameters for the InstantiateVnf operation
      required: false
    scale:
      type: toska.datatypes.nfv.VnfScaleOperationConfiguration
      description: Configuration parameters for the ScaleVnf operation
      required: false
    scale_to_level:
      type: toska.datatypes.nfv.VnfScaleToLevelOperationConfiguration
      description: Configuration parameters for the ScaleVnfToLevel operation
      required: false
    change_flavour:

```

```

    type: toska.datatypes.nfv.VnfChangeFlavourOperationConfiguration
    description: Configuration parameters for the changeVnfFlavourOpConfig
operation
    required: false
    heal:
      type: toska.datatypes.nfv.VnfHealOperationConfiguration
      description: Configuration parameters for the HealVnf operation
      required: false
    terminate:
      type: toska.datatypes.nfv.VnfTerminateOperationConfiguration
      description: Configuration parameters for the TerminateVnf operation
      required: false
    operate:
      type: toska.datatypes.nfv.VnfOperateOperationConfiguration
      description: Configuration parameters for the OperateVnf operation
      required: false
    change_ext_connectivity:
      type: toska.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration
      description: Configuration parameters for the changeExtVnfConnectivityOpConfig
operation
    required: false

```

6.2.20.4 Examples

None.

6.2.20.5 Additional Requirements

None.

6.2.21 toska.datatypes.nfv.VnfInstantiateOperationConfiguration

6.2.21.1 Description

The VnfInstantiateOperationConfiguration data type represents information that affect the invocation of the InstantiateVnf operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.21.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.21.1-1: Type name, shorthand, and URI

Shorthand Name	VnfInstantiateOperationConfiguration
Type Qualified Name	toscanfv:VnfInstantiateOperationConfiguration
Type URI	tosca.datatypes.nfv.VnfInstantiateOperationConfiguration

6.2.21.2 Properties

None.

6.2.21.3 Definition

The syntax of the VnfInstantiateOperationConfiguration data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfInstantiateOperationConfiguration:
  derived_from: toska.datatypes.Root
  description: represents information that affect the invocation of the
InstantiateVnf operation.
  # properties:

```

6.2.21.4 Examples

See clause 6.8.1.9.

6.2.21.5 Additional Requirements

None.

6.2.22 `tosca.datatypes.nfv.VnfScaleOperationConfiguration`

6.2.22.1 Description

`VnfScaleOperationConfiguration` represents information that affect the invocation of the `ScaleVnf` operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.22.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.22.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfScaleOperationConfiguration</code>
Type Qualified Name	<code>toscanfv:VnfScaleOperationConfiguration</code>
Type URI	<code>tosca.datatypes.nfv.VnfScaleOperationConfiguration</code>

6.2.22.2 Properties

The properties of the `VnfScaleOperationConfiguration` data type shall comply with the provisions set out in table 6.2.22.2-1.

Table 6.2.22.2-1: Properties

Name	Required	Type	Constraints	Description
<code>scaling_by_more_than_one_step_supported</code>	no	boolean		Signals whether passing a value larger than one in the <code>numScalingSteps</code> parameter of the <code>ScaleVnf</code> operation is supported by this VNF. Default is <code>FALSE</code> , i.e. "not supported".

6.2.22.3 Definition

The syntax of the `VnfScaleOperationConfiguration` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfScaleOperationConfiguration:
  derived_from: toasca.datatypes.Root
  description: Represents information that affect the invocation of the ScaleVnf
operation
  properties:
    scaling_by_more_than_one_step_supported:
      type: boolean
      description: Signals whether passing a value larger than one in the
numScalingSteps parameter of the ScaleVnf operation is supported by this VNF.
      required: false
      default: false
```

6.2.22.4 Examples

See clause 6.8.1.9.

6.2.22.5 Additional Requirements

None.

6.2.23 `tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration`

6.2.23.1 Description

The `VnfScaleToLevelOperationConfiguration` data type represents information that affect the invocation of the `ScaleVnfToLevel` operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.23.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.23.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfScaleToLevelOperationConfiguration</code>
Type Qualified Name	<code>toscanfv:VnfScaleToLevelOperationConfiguration</code>
Type URI	<code>tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration</code>

6.2.23.2 Properties

The properties of the `VnfScaleToLevelOperationConfiguration` data type shall comply with the provisions set out in table 6.2.23.2-1.

Table 6.2.23.2-1: Properties

Name	Required	Type	Constraints	Description
<code>arbitrary_target_levels_supported</code>	yes	boolean		Signals whether scaling according to the parameter "scaleInfo" is supported by this VNF.

6.2.23.3 Definition

The syntax of the `VnfScaleToLevelOperationConfiguration` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration:
  derived_from: toska.datatypes.Root
  description: represents information that affect the invocation of the
ScaleVnfToLevel operation
  properties:
    arbitrary_target_levels_supported:
      type: boolean
      description: Signals whether scaling according to the parameter "scaleInfo" is
supported by this VNF
      required: true
```

6.2.23.4 Examples

See clause 6.8.1.9.

6.2.23.5 Additional Requirements

None.

6.2.24 tosca.datatypes.nfv.VnfHealOperationConfiguration

6.2.24.1 Description

The VnfHealOperationConfiguration data type represents information that affect the invocation of the HealVnf operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.24.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.24.1-1: Type name, shorthand, and URI

Shorthand Name	VnfHealOperationConfiguration
Type Qualified Name	toscanfv:VnfHealOperationConfiguration
Type URI	tosca.datatypes.nfv.VnfHealOperationConfiguration

6.2.24.2 Properties

The properties of the VnfHealOperationConfiguration data type shall comply with the provisions set out in table 6.2.24.2-1.

Table 6.2.24.2-1: Properties

Name	Required	Type	Constraints	Description
causes	no	list of string		Supported "cause" parameter values.

6.2.24.3 Definition

The syntax of the VnfHealOperationConfiguration data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfHealOperationConfiguration:
  derived_from: tosca.datatypes.Root
  description: represents information that affect the invocation of the HealVnf
operation
  properties:
    causes:
      type: list
      description: Supported "cause" parameter values
      required: false
      entry_schema:
        type: string
```

6.2.24.4 Examples

See clause 6.8.1.9.

6.2.24.5 Additional Requirements

None.

6.2.25 tosca.datatypes.nfv.VnfTerminateOperationConfiguration

6.2.25.1 Description

The VnfTerminateOperationConfiguration data type represents information that affect the invocation of the TerminateVnf, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.25.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.25.1-1: Type name, shorthand, and URI

Shorthand Name	VnfTerminateOperationConfiguration
Type Qualified Name	toscanfv:VnfTerminateOperationConfiguration
Type URI	tosca.datatypes.nfv.VnfTerminateOperationConfiguration

6.2.25.2 Properties

The properties of the VnfTerminateOperationConfiguration data type shall comply with the provisions set out in table 6.2.25.2-1.

Table 6.2.25.2-1: Properties

Name	Required	Type	Constraints	Description
min_graceful_termination_timeout	yes	scalar-unit.time		Minimum timeout value for graceful termination of a VNF instance.
max_recommended_graceful_termination_timeout	no	scalar-unit.time		Maximum recommended timeout value that can be needed to gracefully terminate a VNF instance of a particular type under certain conditions, such as maximum load condition. This is provided by VNF provider as information for the operator facilitating the selection of optimal timeout value. This value is not used as constraint.

6.2.25.3 Definition

The syntax of the VnfTerminateOperationConfiguration data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfTerminateOperationConfiguration:
  derived_from: toscan.datatypes.Root
  description: represents information that affect the invocation of the TerminateVnf
  properties:
    min_graceful_termination_timeout:
      type: scalar-unit.time
      description: Minimum timeout value for graceful termination of a VNF instance
      required: true
    max_recommended_graceful_termination_timeout:
      type: scalar-unit.time
      description: Maximum recommended timeout value that can be needed to
  gracefully terminate a VNF instance of a particular type under certain conditions,
  such as maximum load condition. This is provided by VNF provider as information for
  the operator facilitating the selection of optimal timeout value. This value is not
  used as constraint
      required: false

```

6.2.25.4 Examples

None.

6.2.25.5 Additional Requirements

None.

6.2.26 toska.datatypes.nfv.VnfOperateOperationConfiguration

6.2.26.1 Description

The VnfOperateOperationConfiguration data type represents information that affect the invocation of the OperateVnf operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.26.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.26.1-1: Type name, shorthand, and URI

Shorthand Name	VnfOperateOperationConfiguration
Type Qualified Name	toscanfv:VnfOperateOperationConfiguration
Type URI	tosca.datatypes.nfv.VnfOperateOperationConfiguration

6.2.26.2 Properties

The properties of the VnfOperateOperationConfiguration data type shall comply with the provisions set out in table 6.2.26.2-1.

Table 6.2.26.2-1: Properties

Name	Required	Type	Constraints	Description
min_graceful_stop_timeout	yes	scalar-unit.time		Minimum timeout value for graceful stop of a VNF instance.
max_recommended_graceful_stop_timeout	no	scalar-unit.time		Maximum recommended timeout value that can be needed to gracefully stop a VNF instance of a particular type under certain conditions, such as maximum load condition. This is provided by VNF provider as information for the operator facilitating the selection of optimal timeout value. This value is not used as constraint.

6.2.26.3 Definition

The syntax of the VnfOperateOperationConfiguration data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfOperateOperationConfiguration:
  derived_from: toska.datatypes.Root
  description: represents information that affect the invocation of the OperateVnf
operation
  properties:
    min_graceful_stop_timeout:
      type: scalar-unit.time
      description: Minimum timeout value for graceful stop of a VNF instance
      required: true
    max_recommended_graceful_stop_timeout:
      type: scalar-unit.time
      description: Maximum recommended timeout value that can be needed to
gracefully stop a VNF instance of a particular type under certain conditions, such as
maximum load condition. This is provided by VNF provider as information for the
operator facilitating the selection of optimal timeout value. This value is not used
as constraint
      required: false
```

6.2.26.4 Examples

None.

6.2.26.5 Additional Requirements

None.

6.2.27 `tosca.datatypes.nfv.ScaleInfo`

6.2.27.1 Description

The `scaleInfo` data type indicates for a given `scaleAspect` the corresponding `scaleLevel`. Table 6.2.27.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.27.1-1: Type name, shorthand, and URI

Shorthand Name	<code>ScaleInfo</code>
Type Qualified Name	<code>toscanfv:ScaleInfo</code>
Type URI	<code>tosca.datatypes.nfv.ScaleInfo</code>

6.2.27.2 Properties

The properties of the `ScaleInfo` data type shall comply with the provisions set out in table 6.2.27.2-1.

Table 6.2.27.2-1: Properties

Name	Required	Type	Constraints	Description
<code>scale_level</code>	yes	integer	<code>greater_or_equal : 0</code>	The scale level for a particular aspect.

6.2.27.3 Definition

The syntax of the `ScaleInfo` data type shall comply with the following definition:

```
tosca.datatypes.nfv.ScaleInfo:
  derived_from: toska.datatypes.Root
  description: Indicates for a given scaleAspect the corresponding scaleLevel
  properties:
    scale_level:
      type: integer
      description: The scale level for a particular aspect
      required: true
      constraints:
        - greater_or_equal: 0
```

6.2.27.4 Examples

See clause 6.8.1.9.

6.2.27.5 Additional Requirements

None.

6.2.28 `tosca.datatypes.nfv.ScalingAspect`

6.2.28.1 Description

The `ScalingAspect` data type describes the details of an aspect used for horizontal scaling. Table 6.2.28.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.28.1-1: Type name, shorthand, and URI

Shorthand Name	ScalingAspect
Type Qualified Name	toscanfv:ScalingAspect
Type URI	tosca.datatypes.nfv.ScalingAspect

6.2.28.2 Properties

The properties of the ScalingAspect data type shall comply with the provisions set out in table 6.2.28.2-1.

Table 6.2.28.2-1: Properties

Name	Required	Type	Constraints	Description
name	yes	string		Human readable name of the aspect.
description	yes	string		Human readable description of the aspect.
max_scale_level	yes	integer	positiveInteger	Total number of scaling steps that can be applied w.r.t. this aspect. The value of this property corresponds to the number of scaling steps can be applied to this aspect when scaling it from the minimum scale level (i.e. 0) to the maximum scale level defined by this property.
step_deltas	no	list of string		List of scaling deltas to be applied for the different subsequent scaling steps of this aspect. The first entry in the array shall correspond to the first scaling step (between scale levels 0 to 1) and the last entry in the array shall correspond to the last scaling step (between maxScaleLevel-1 and maxScaleLevel).

6.2.28.3 Definition

The syntax of the ScalingAspect data type shall comply with the following definition:

```
tosca.datatypes.nfv.ScalingAspect:
  derived_from: toscanfv:Root
  properties:
    name:
      type: string
      required: true
    description:
      type: string
      required: true
    max_scale_level:
      type: integer # positiveInteger
      required: true
      constraints:
        - greater_or_equal: 0
    step_deltas:
      type: list
      required: false
    entry_schema:
      type: string # Identifier
```

6.2.28.4 Examples

See clause A.6.

6.2.28.5 Additional Requirements

None.

6.2.29 `tosca.datatypes.nfv.LinkBitrateRequirements`

6.2.29.1 Description

The `LinkBitrateRequirements` data type describes the requirements in terms of bitrate for a virtual link. Table 6.2.29.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.29.1-1: Type name, shorthand, and URI

Shorthand Name	<code>LinkBitrateRequirements</code>
Type Qualified Name	<code>toscanfv:LinkBitrateRequirements</code>
Type URI	<code>tosca.datatypes.nfv.LinkBitrateRequirements</code>

6.2.29.2 Properties

The properties of the `LinkBitrateRequirements` data type shall comply with the provisions set out in table 6.2.29.2-1.

Table 6.2.29.2-1: Properties

Name	Required	Type	Constraints	Description
root	yes	integer	greater_or_equal: 0	Specifies the throughput requirement in bits per second of the link (e.g. bitrate of E-Line, root bitrate of E-Tree, aggregate capacity of E-LAN).
leaf	no	integer	greater_or_equal: 0	Specifies the throughput requirement in bits per second of leaf connections to the link when applicable to the connectivity type (e.g. for E-Tree and E-LAN branches).

6.2.29.3 Definition

The syntax of the `LinkBitrateRequirements` data type shall comply with the following definition:

```
tosca.datatypes.nfv.LinkBitrateRequirements:
  derived_from: toasca.datatypes.Root
  description: describes the requirements in terms of bitrate for a virtual link
  properties:
    root:
      type: integer # in bits per second
      description: Specifies the throughput requirement in bits per second of the
link (e.g. bitrate of E-Line, root bitrate of E-Tree, aggregate capacity of E-LAN).
      required: true
      constraints:
        - greater_or_equal: 0
    leaf:
      type: integer # in bits per second
      description: Specifies the throughput requirement in bits per second of leaf
connections to the link when applicable to the connectivity type (e.g. for E-Tree and
E LAN branches).
      required: false
      constraints:
        - greater_or_equal: 0
```

6.2.29.4 Examples

None.

6.2.29.5 Additional Requirements

None.

6.2.30 tosca.datatypes.nfv.Qos

6.2.30.1 Description

The QoS describes QoS data type a given VL used in a VNF deployment flavour. Table 6.2.30.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.30.1-1: Type name, shorthand, and URI

Shorthand Name	Qos
Type Qualified Name	toscanfv:Qos
Type URI	tosca.datatypes.nfv.Qos

6.2.30.2 Properties

The properties of the Qos data type shall comply with the provisions set out in table 6.2.30.2-1.

Table 6.2.30.2-1: Properties

Name	Required	Type	Constraints	Description
latency	yes	scalar-unit.time	greater_than: 0 s	Specifies the maximum latency.
packet_delay_variation	yes	scalar-unit.time		Specifies the maximum jitter.
packet_loss_ratio	no	float	in_range: [0,1]	Specifies the maximum packet loss ratio.

6.2.30.3 Definition

The syntax of the Qos data type shall comply with the following definition:

```
tosca.datatypes.nfv.Qos:
  derived_from: tosca.datatypes.Root
  description: describes QoS data for a given VL used in a VNF deployment flavour
  properties:
    latency:
      type: scalar-unit.time #Number
      description: Specifies the maximum latency
      required: true
      constraints:
        - greater_than: 0 s
    packet_delay_variation:
      type: scalar-unit.time #Number
      description: Specifies the maximum jitter
      required: true
      constraints:
        - greater_or_equal: 0 s
    packet_loss_ratio:
      type: float
      description: Specifies the maximum packet loss ratio
      required: false
```

```
constraints:
  - in_range: [ 0.0, 1.0 ]
```

6.2.30.4 Examples

None.

6.2.30.5 Additional Requirements

None.

6.2.31 `tosca.datatypes.nfv.VnfConfigurableProperties`

6.2.31.1 Description

The `VnfConfigurableProperties` data type describes configurable properties for a given VNF (e.g. related to auto scaling and auto healing).

Table 6.2.31.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.31.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfConfigurableProperties</code>
Type Qualified Name	<code>toscanfv:VnfConfigurableProperties</code>
Type URI	<code>tosca.datatypes.nfv.VnfConfigurableProperties</code>

6.2.31.2 Properties

The properties of the `VnfConfigurableProperties` data type shall comply with the provisions set out in table 6.2.31.2-1.

Table 6.2.31.2-1: Properties

Name	Required	Type	Constraints	Description
<code>is_autoscale_enabled</code>	no	boolean		It permits to enable (TRUE)/disable (FALSE) the auto-scaling functionality. If the properties is not present for configuring, then VNF property is not supported
<code>is_autoheal_enabled</code>	no	boolean		It permits to enable (TRUE)/disable (FALSE) the auto-healing functionality. If the properties is not present for configuring, then VNF property is not supported
<code>additional_configurable_property</code>	no	<code>tosca.datatypes.nfv.VnfAdditionalConfigurableProperties</code>		It provides VNF specific configurable properties that can be modified using the <code>ModifyVnfInfo</code> operation

6.2.31.3 Definition

The syntax of the `VnfConfigurableProperties` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfConfigurableProperties:
  derived_from: toasca.datatypes.Root
  description: indicates configuration properties for a given VNF (e.g. related to
  auto scaling and auto healing).
  properties:
```

```

is_autoscale_enabled:
  type: boolean
  description: It permits to enable (TRUE)/disable (FALSE) the auto-scaling
functionality. If the properties is not present for configuring, then VNF property is
not supported
  required: false
is_autoheal_enabled:
  type: boolean
  description: It permits to enable (TRUE)/disable (FALSE) the auto-healing
functionality. If the properties is not present for configuring, then VNF property is
not supported
  required: false
# additional_configurable_properties:
# description: It provides VNF specific configurable properties that can be
# modified using the ModifyVnfInfo operation
# required: false
# type: toasca.datatypes.nfv.VnfAdditionalConfigurableProperties
# derived types are expected to introduce
# additional_configurable_properties with its type derived from
# toasca.datatypes.nfv.VnfAdditionalConfigurableProperties

```

6.2.31.4 Examples

Example definition of configurable properties.

```

tosca_definitions_version: toasca_simple_yaml_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: toasca.nodes.nfv.VNF
    properties:
      flavour_id:
        constraints:
          - valid_values: [ simple, complex ]
      configurable_properties:
        type: MyCompany.datatypes.nfv.VnfConfigurableProperties

data_types:
  Mycompany.datatypes.nfv.VnfConfigurableProperties:
    derived_from: toasca.datatypes.nfv.VnfConfigurableProperties
    properties:
      additional_configurable_properties:
        type: MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties

  MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties:
    derived_from: toasca.datatypes.nfv.VnfAdditionalConfigurableProperties
    properties:
      name_prefix_in_vim:
        type: string
        required: false
      dns_server:
        type: string
        required: true

```

6.2.31.5 Additional Requirements

None.

6.2.32 `tosca.datatypes.nfv.VnfAdditionalConfigurableProperties`

6.2.32.1 Description

The `VnfAdditionalConfigurableProperties` data type is an empty base type for deriving data types for describing additional configurable properties for a given VNF. Table 6.2.32.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.32.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfAdditionalConfigurableProperties</code>
Type Qualified Name	<code>toscanfv:VnfAdditionalConfigurableProperties</code>
Type URI	<code>tosca.datatypes.nfv.VnfAdditionalConfigurableProperties</code>

6.2.32.2 Properties

None.

6.2.32.3 Definition

The syntax of the `VnfAdditionalConfigurableProperties` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfAdditionalConfigurableProperties:
  derived_from: toasca.datatypes.Root
  description: is an empty base type for deriving data types for describing
  additional configurable properties for a given VNF
```

6.2.32.4 Examples

See clause 6.2.31.4.

6.2.32.5 Additional Requirements

None.

6.2.33 `tosca.datatypes.nfv.VnfInfoModifiableAttributes`

6.2.33.1 Description

The `VnfInfoModifiableAttributes` data type describes VNF-specific extension and metadata for a given VNF. Table 6.2.33.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.33.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfInfoModifiableAttributes</code>
Type Qualified Name	<code>toscanfv:VnfInfoModifiableAttributes</code>
Type URI	<code>tosca.datatypes.nfv.VnfInfoModifiableAttributes</code>

6.2.33.2 Properties

The properties of the `VnfInfoModifiableAttributes` data type shall comply with the provisions set out in table 6.2.33.2-1.

Table 6.2.33.2-1: Properties

Name	Required	Type	Constraints	Description
extensions	no	tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions		"Extension" properties of VnfInfo that are writeable.
metadata	no	tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata		"Metadata" properties of VnfInfo that are writeable.

6.2.33.3 Definition

The syntax of the VnfInfoModifiableAttributes data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfInfoModifiableAttributes:
  derived_from: toska.datatypes.Root
  description: Describes VNF-specific extension and metadata for a given VNF
  #properties:
    #extensions:
      #type: toska.datatypes.nfv.VnfInfoModifiableAttributesExtensions
      #description: "Extension" properties of VnfInfo that are writeable
      #required: false
      # derived types are expected to introduce
      # extensions with its type derived from
      # toska.datatypes.nfv.VnfInfoModifiableAttributesExtensions
    #metadata:
      #type: toska.datatypes.nfv.VnfInfoModifiableAttributesMetadata
      #description: "Metadata" properties of VnfInfo that are writeable
      #required: false
      # derived types are expected to introduce
      # metadata with its type derived from
      # toska.datatypes.nfv.VnfInfoModifiableAttributesMetadata

```

6.2.33.4 Examples

See clause 6.8.1.9.

6.2.33.5 Additional Requirements

None.

6.2.34 toska.datatypes.nfv.VnfInfoModifiableAttributesExtensions

6.2.34.1 Description

The VnfInfoModifiableAttributesExtensions data type is an empty base type for deriving data types for describing VNF-specific extension. Table 6.2.34.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.34.1-1: Type name, shorthand, and URI

Shorthand Name	VnfInfoModifiableAttributesExtensions
Type Qualified Name	toscanfv:VnfInfoModifiableAttributesExtensions
Type URI	tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions

6.2.34.2 Properties

None.

6.2.34.3 Definition

The syntax of the VnfInfoModifiableAttributesExtensions data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions:
  derived_from: toska.datatypes.Root
  description: is an empty base type for deriving data types for describing VNF-
specific extension
```

6.2.34.4 Examples

See clause 6.8.1.9.

6.2.34.5 Additional Requirements

None.

6.2.35 toska.datatypes.nfv.VnfInfoModifiableAttributesMetadata

6.2.35.1 Description

The VnfInfoModifiableAttributesMetadata data type is an empty base type for deriving data types for describing VNF-specific metadata. Table 6.2.35.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.35.1-1: Type name, shorthand, and URI

Shorthand Name	VnfInfoModifiableAttributesMetadata
Type Qualified Name	toscanfv:VnfInfoModifiableAttributesMetadata
Type URI	tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata

6.2.35.2 Properties

None.

6.2.35.3 Definition

The syntax of the VnfInfoModifiableAttributesMetadata data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata:
  derived_from: toska.datatypes.Root
  description: is an empty base type for deriving data types for describing VNF-
specific metadata
```

6.2.35.4 Examples

Example metadata definition:

```
tosca_definitions_version: toska_simple_yaml_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
```

```

derived_from: tosca.nodes.nfv.VNF
properties:
  flavour_id:
    constraints:
      - valid_values: [ simple, complex ]
  modifiable_attributes:
    type: mycompany.datatypes.nfv.VnfInfoModifiableAttributes

data_types:
  mycompany.datatypes.nfv.VnfInfoModifiableAttributes:
    derived_from: tosca.datatypes.nfv.VnfInfoModifiableAttributes
    properties:
      metadata:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesMetadata

  mycompany.datatypes.nfv.VnfInfoModifiableAttributesMetadata:
    derived_from: tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata
    properties:
      metadata_key_1:
        type: string
        required: false
      metadata_key_2:
        type: string
        required: false

```

6.2.35.5 Additional Requirements

None.

6.2.36 tosca.datatypes.nfv.CpProtocolData

6.2.36.1 Description

The CpProtocolData data type describes and associates the protocol layer that a CP uses together with other protocol and connection point information. Table 6.2.36.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.36.1-1: Type name, shorthand, and URI

Shorthand Name	CpProtocolData
Type Qualified Name	toscanfv:CpProtocolData
Type URI	tosca.datatypes.nfv.CpProtocolData

6.2.36.2 Properties

The properties of the CpProtocolData data type shall comply with the provisions set out in table 6.2.36.2-1.

Table 6.2.36.2-1: Properties

Name	Required	Type	Constraints	Description
associated_layer_protocol	yes	string	Valid values: ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire	One of the values of the property layer_protocols of the CP.
address_data	no	list of tosca.datatypes.nfv.AddressData		Provides information on the addresses to be assigned to the CP.

6.2.36.3 Definition

The syntax of the CpProtocolData data type shall comply with the following definition:

```

tosca.datatypes.nfv.CpProtocolData:
  derived_from: tosca.datatypes.Root
  description: Describes and associates the protocol layer that a CP uses together
with other protocol and connection point information
.
  properties:
    associated_layer_protocol:
      type: string
      required: true
      description: One of the values of the property layer_protocols of the CP
      constraints:
        - valid_values: [ ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire ]
    address_data:
      type: list
      description: Provides information on the addresses to be assigned to the CP
    entry_schema:
      type: tosca.datatypes.nfv.AddressData
      required: false

```

6.2.36.4 Examples

None.

6.2.36.5 Additional Requirements

None.

6.2.37 tosca.datatypes.nfv.LogicalNodeData

6.2.37.1 Description

The LogicalNodeData data type describes compute, memory and I/O requirements associated with a particular VDU. Table 6.2.37.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.37.1-1: Type name, shorthand, and URI

Shorthand Name	LogicalNodeData
Type Qualified Name	toscanfv:LogicalNodeData
Type URI	tosca.datatypes.nfv.LogicalNodeData

6.2.37.2 Properties

The properties of the LogicalNodeData data type shall comply with the provisions set out in table 6.2.37.2-1.

Table 6.2.37.2-1: Properties

Name	Required	Type	Constraints	Description
logical_node_requirements	no	map of string		<p>The logical node-level compute, memory and I/O requirements. A map of strings that contains a set of key-value pairs that describes hardware platform specific deployment requirements, including the number of CPU cores on this logical node, a memory configuration specific to a logical node or a requirement related to the association of an I/O device with the logical node.</p> <p>More information regarding the usage of this property is available at: https://register.etsi.org</p>

6.2.37.3 Definition

The syntax of the LogicalNodeData data type shall comply with the following definition:

```

tosca.datatypes.nfv.LogicalNodeData:
  derived_from: toska.datatypes.Root
  description: Describes compute, memory and I/O requirements associated with a
particular VDU.
  properties:
    logical_node_requirements:
      type: map
      description: The logical node-level compute, memory and I/O requirements. A
map of strings that contains a set of key-value pairs that describes hardware
platform specific deployment requirements, including the number of CPU cores on this
logical node, a memory configuration specific to a logical node or a requirement
related to the association of an I/O device with the logical node.
      required: false
      entry_schema:
        type: string

```

6.2.37.4 Examples

None.

6.2.37.5 Additional Requirements

None.

6.2.38 toska.datatypes.nfv.SwImageData

6.2.38.1 Description

The SwImageData data type describes information related to a software image artifact. Table 6.2.38.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.38.1-1: Type name, shorthand, and URI

Shorthand Name	SwImageData
Type Qualified Name	toscanfv:SwImageData
Type URI	tosca.datatypes.nfv.SwImageData

6.2.38.2 Properties

The properties of the SwImageData data type shall comply with the provisions set out in table 6.2.38.2-1.

Table 6.2.38.2-1: 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	Valid values: [aki, ami, ari, bare, docker, ova, ovf]	The container format describes the container file format in which software image is provided. Description of valid values: aki: a kernel image ami: a machine image ari: a ramdisk image bare: the image does not have a container or metadata envelope docker: docker container format ova: OVF package in a tarfile ovf: OVF container format Future versions of the present document may extend the list of possible values.
disk_format	yes	string	Valid values: [aki, ami, ari, iso, qcow2, raw, vdi, vhd, vhdx, vmdk]	The disk format of a software image is the format of the underlying disk image. Description of valid values: [aki: a kernel image ami: a machine image ari: a ramdisk image iso: an archive format for the data contents of an optical disc, such as CD-ROM qcow2: a common disk image format, which can expand dynamically and supports copy on write raw: an unstructured disk image format vdi: a common disk image format vhd: a common disk image format vhdx: enhanced version of VHD format vmdk: a common disk image format Future versions of the present document may extend the list of possible values.
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.
operating_system	no	string		Identifies the operating system used in the software image.
supported_virtualisation_environments	no	list of string		Identifies the virtualisation environments (e.g. hypervisor) compatible with this software image.

6.2.38.3 Definition

The syntax of the SwImageData data type shall comply with the following definition:

```

tosca.datatypes.nfv.SwImageData:
  derived_from: toska.datatypes.Root
  description: describes information related to a software image artifact
  properties: # in SOL001 v0.8.0: "properties or metadata:"
    name:
      type: string
      description: Name of this software image
      required: true
    version:
      type: string
      description: Version of this software image
      required: true
    checksum:
      type: string
      description: Checksum of the software image file
      required: true
    container_format:
      type: string
      description: The container format describes the container file format in which
software image is provided
      required: true
      constraints:
        - valid_values: [ aki, ami, ari, bare, docker, ova, ovf ]
    disk_format:
      type: string
      description: The disk format of a software image is the format of the
underlying disk image
      required: true
      constraints:
        - valid_values: [ aki, ami, ari, iso, qcow2, raw, vdi, vhd, vhdx, vmdk ]
    min_disk:
      type: scalar-unit.size # Number
      description: The minimal disk size requirement for this software image
      required: true
    min_ram:
      type: scalar-unit.size # Number
      description: The minimal RAM requirement for this software image
      required: false
    size:
      type: scalar-unit.size # Number
      description: The size of this software image
      required: true
    operating_system:
      type: string
      description: Identifies the operating system used in the software image
      required: false
    supported_virtualisation_environments:
      type: list
      description: Identifies the virtualisation environments (e.g. hypervisor)
compatible with this software image
      required: false
    entry_schema:
      type: string

```

6.2.38.4 Examples

None.

6.2.38.5 Additional Requirements

None.

6.2.39 `tosca.datatypes.nfv.VirtualBlockStorageData`

6.2.39.1 Description

The `VirtualBlockStorageData` data type describes block storage requirements associated with compute resources in a particular VDU, either as a local disk or as virtual attached storage. Table 6.2.39.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.39.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualBlockStorageData</code>
Type Qualified Name	<code>toscanfv:VirtualBlockStorageData</code>
Type URI	<code>tosca.datatypes.nfv.VirtualBlockStorageData</code>

6.2.39.2 Properties

The properties of the `VirtualBlockStorageData` data type shall comply with the provisions set out in table 6.2.39.2-1.

Table 6.2.39.2-1: Properties

Name	Required	Type	Constraints	Description
<code>size_of_storage</code>	yes	scalar-unit.size		Size of virtualised storage resource
<code>vdu_storage_requirements</code>	no	map of string		The hardware platform specific storage requirements. A map of strings that contains a set of key-value pairs that represents the hardware platform specific storage deployment requirements More information regarding the usage of this property is available at: https://register.etsi.org
<code>rdma_enabled</code>	no	boolean	default: false	Indicate if the storage support RDMA

6.2.39.3 Definition

The syntax of the `VirtualBlockStorageData` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VirtualBlockStorageData:
  derived_from: toska.datatypes.Root
  description: VirtualBlockStorageData describes block storage requirements
  associated with compute resources in a particular VDU, either as a local disk or as
  virtual attached storage
  properties:
    size_of_storage:
      type: scalar-unit.size
      description: Size of virtualised storage resource
      required: true
    vdu_storage_requirements:
      type: map
```

```

description: The hardware platform specific storage requirements. A map of
strings that contains a set of key-value pairs that represents the hardware platform
specific storage deployment requirements
required: false
entry_schema:
  type: string
rdma_enabled:
  type: boolean
description: Indicates if the storage support RDMA
required: false
default: false

```

6.2.39.4 Examples

None.

6.2.39.5 Additional Requirements

None.

6.2.40 `tosca.datatypes.nfv.VirtualObjectStorageData`

6.2.40.1 Description

The `VirtualObjectStorageData` data type describes object storage requirements associated with compute resources in a particular VDU. Table 6.2.40.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.40.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualObjectStorageData</code>
Type Qualified Name	<code>toscanfv:VirtualObjectStorageData</code>
Type URI	<code>tosca.datatypes.nfv.VirtualObjectStorageData</code>

6.2.40.2 Properties

The properties of the `VirtualObjectStorageData` data type shall comply with the provisions set out in table 6.2.40.2-1.

Table 6.2.40.2-1: Properties

Name	Required	Type	Constraints	Description
<code>max_size_of_storage</code>	no	<code>scalar-unit.size</code>		Maximum size of virtualized storage resource

6.2.40.3 Definition

The syntax of the `VirtualObjectStorageData` data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualObjectStorageData:
  derived_from: toasca.datatypes.Root
  description: VirtualObjectStorageData describes object storage requirements
associated with compute resources in a particular VDU
  properties:
    max_size_of_storage:
      type: scalar-unit.size
      description: Maximum size of virtualized storage resource

```


required: false

6.2.40.4 Examples

None.

6.2.40.5 Additional Requirements

None.

6.2.41 `tosca.datatypes.nfv.VirtualFileStorageData`

6.2.41.1 Description

The `VirtualObjectFileData` data type describes file storage requirements associated with compute resources in a particular VDU. Table 6.2.41.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.41.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualObjectFileData</code>
Type Qualified Name	<code>toscanfv:VirtualFileStorageData</code>
Type URI	<code>tosca.datatypes.nfv.VirtualFileStorageData</code>

6.2.41.2 Properties

The properties of the `VirtualFileStorageData` data type shall comply with the provisions set out in table 6.2.41.2-1.

Table 6.2.41.2-1: Properties

Name	Required	Type	Constraints	Description
<code>size_of_storage</code>	yes	<code>scalar-unit.size</code>		Maximum size of virtualized storage resource
<code>file_system_protocol</code>	yes	<code>string</code>		The shared file system protocol (e.g. NFS, CIFS).

6.2.41.3 Definition

The syntax of the `VirtualFileStorageData` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VirtualFileStorageData:
  derived_from: toasca.datatypes.Root
  description: VirtualFileStorageData describes file storage requirements
  associated with compute resources in a particular VDU
  properties:
    size_of_storage:
      type: scalar-unit.size
      description: Size of virtualized storage resource
      required: true
    file_system_protocol:
      type: string
      description: The shared file system protocol (e.g. NFS, CIFS)
      required: true
```

6.2.41.4 Examples

None.

6.2.41.5 Additional Requirements

None.

6.2.42 `tosca.datatypes.nfv.VirtualLinkBitrateLevel`

6.2.42.1 Description

The `VirtualLinkBitrateLevel` data type describes bitrate requirements applicable to the virtual link instantiated from a particular `VnfVirtualLink`. Table 6.2.42.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.42.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualLinkBitrateLevel</code>
Type Qualified Name	<code>toscanfv:VirtualLinkBitrateLevel</code>
Type URI	<code>tosca.datatypes.nfv.VirtualLinkBitrateLevel</code>

6.2.42.2 Properties

The properties of the `VirtualLinkBitrateLevel` data type shall comply with the provisions set out in table 6.2.42.2-1.

Table 6.2.42.2-1: Properties

Name	Required	Type	Constraints	Description
<code>bitrate_requirements</code>	yes	<code>tosca.datatypes.nfv.LinkBitrateRequirements</code>		Virtual link bitrate requirements for an instantiation level or bitrate delta for a scaling step.

6.2.42.3 Definition

The syntax of the `VirtualLinkBitrateLevel` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VirtualLinkBitrateLevel:
  derived_from: toasca.datatypes.Root
  description: Describes bitrate requirements applicable to the virtual link
instantiated from a particular VnfVirtualLink
  properties:
    bitrate_requirements:
      type: toasca.datatypes.nfv.LinkBitrateRequirements
      description: Virtual link bitrate requirements for an instantiation level or
bitrate delta for a scaling step
      required: true
```

6.2.42.4 Examples

See clause A.6.

6.2.42.5 Additional Requirements

None.

6.2.43 tosca.datatypes.nfv.VnfOperationAdditionalParameters

6.2.43.1 Description

The VnfOperationAdditionalParameters data type is an empty base type for deriving data type for describing VNF-specific parameters to be passed when invoking lifecycle management operations as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.43.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.43.1-1: Type name, shorthand, and URI

Shorthand Name	VnfOperationAdditionalParameters
Type Qualified Name	toscanfv:VnfOperationAdditionalParameters
Type URI	tosca.datatypes.nfv.VnfOperationAdditionalParameters

6.2.43.2 Properties

None.

6.2.43.3 Definition

The syntax of the VnfOperationAdditionalParameters data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfOperationAdditionalParameters:
  derived_from: tosca.datatypes.Root
  description: Is an empty base type for deriving data type for describing VNF-
specific parameters to be passed when invoking lifecycle management operations
  # properties:
```

6.2.43.4 Examples

```
tosca_definitions_version: tosca_simple_yaml_1_2

node_types:
  MyCompany.nodes.nfv.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ..
    interfaces:
      Vnflcm:
        instantiate:
          inputs:
            additional_parameters:
              type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters

data_types:
  MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      parameter_1:
        type: string
        required: true
      parameter_2:
        type: string
        required: false
        default: value_2
```

6.2.43.5 Additional Requirements

None.

6.2.44 `tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration`

6.2.44.1 Description

The `VnfChangeFlavourOperationConfiguration` data type represents information that affect the invocation of the `ChangeVnfFlavour` operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.44.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.44.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfChangeFlavourOperationConfiguration</code>
Type Qualified Name	<code>toscanfv:VnfChangeFlavourOperationConfiguration</code>
Type URI	<code>tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration</code>

6.2.44.2 Properties

None.

6.2.44.3 Definition

The syntax of the `VnfChangeFlavourOperationConfiguration` data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration:
  derived_from: toasca.datatypes.Root
  description: represents information that affect the invocation of the
ChangeVnfFlavour operation
  # properties:
```

6.2.44.4 Examples

None.

6.2.44.5 Additional Requirements

None.

6.2.45 `tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration`

6.2.45.1 Description

The `VnfChangeExtConnectivityOperationConfiguration` data type represents information that affect the invocation of the `ChangeExtVnfConnectivity` operation, as specified in ETSI GS NFV-IFA 011 [1]. Table 6.2.45.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.45.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfChangeExtConnectivityOperationConfiguration</code>
Type Qualified Name	<code>toscanfv:VnfChangeExtConnectivityOperationConfiguration</code>
Type URI	<code>tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration</code>

6.2.45.2 Properties

None.

6.2.45.3 Definition

The syntax of the VnfChangeExtConnectivityOperationConfiguration data type shall comply with the following definition:

```
tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration:
  derived_from: toasca.datatypes.Root
  description: represents information that affect the invocation of the
ChangeExtVnfConnectivity operation
  # properties:
```

6.2.45.4 Examples

None.

6.2.45.5 Additional Requirements

None.

6.2.46 toasca.datatypes.nfv.VnfMonitoringParameter

6.2.46.1 Description

This data type provides information on virtualised resource related performance metrics applicable to VNF. Table 6.2.46.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.46.1-1: Type name, shorthand, and URI

Shorthand Name	VnfMonitoringParameter
Type Qualified Name	toscanfv:VnfMonitoringParameter
Type URI	tosca.datatypes.nfv.VnfMonitoringParameter

6.2.46.2 Properties

The properties of the VnfMonitoringParameter data type shall comply with the provisions set out in table 6.2.46.2-1.

Table 6.2.46.2-1: Properties

Name	Required	Type	Constraints	Description
name	no	string		Human readable name of the monitoring parameter.
performance_metric	yes	string	valid values: v_cpu_usage_mean_vnf v_cpu_usage_peak_vnf v_memory_usage_mean_vnf v_memory_usage_peak_vnf v_disk_usage_mean_vnf v_disk_usage_peak_vnf. byte_incoming_vnf_ext_cp byte_outgoing_vnf_ext_cp packet_incoming_vnf_ext_cp packet_outgoing_vnf_ext_cp	Identifies the performance metric, according to ETSI GS NFV-IFA 027 [7]. Performance metric values shall be set to the corresponding measurement name defined in clause 7.2 of ETSI GS NFV-IFA 027 [7], without appending a sub-counter.
collection_period	no	scalar-unit.time		Describes the recommended periodicity at which to collect the performance information. The VNFM determines if this parameter is considered. The vendor may provide this information as a guidance for creating PmJobs.

6.2.46.3 Definition

The syntax of the VnfMonitoringParameter data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfMonitoringParameter:
  derived_from: toasca.datatypes.Root
  description: Represents information on virtualised resource related performance
  metrics applicable to the VNF.
  properties:
    name:
      type: string
      description: Human readable name of the monitoring parameter
      required: true
    performance_metric:
      type: string
      description: Identifies the performance metric, according to ETSI GS NFV-IFA 027.
      required: true
    constraints:
      - valid_values: [ v_cpu_usage_mean_vnf, v_cpu_usage_peak_vnf,
v_memory_usage_mean_vnf, v_memory_usage_peak_vnf, v_disk_usage_mean_vnf,
v_disk_usage_peak_vnf, byte_incoming_vnf_ext_cp, byte_outgoing_vnf_ext_cp,

```

```

packet_incoming_vnf_ext_cp, packet_outgoing_vnf_ext_cp ]
  collection_period:
    type: scalar-unit.time
    description: Describes the recommended periodicity at which to collect the
performance information.
    required: false
    constraints:
      - greater_than: 0 s

```

6.2.46.4 Examples

None.

6.2.46.5 Additional Requirements

None.

6.2.47 `tosca.datatypes.nfv.VnfcMonitoringParameter`

6.2.47.1 Description

This data type provides information on virtualised resource related performance metrics applicable to a VNFC. Table 6.2.47.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.47.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfcMonitoringParameter</code>
Type Qualified Name	<code>toscanfv:VnfcMonitoringParameter</code>
Type URI	<code>tosca.datatypes.nfv.VnfcMonitoringParameter</code>

6.2.47.2 Properties

The properties of the `VnfcMonitoringParameter` data type shall comply with the provisions set out in table 6.2.47.2-1.

Table 6.2.47.2-1: Properties

Name	Required	Type	Constraints	Description
name	no	string		Human readable name of the monitoring parameter.
performance_metric	yes	string	valid values: v_cpu_usage_mean_vnf v_cpu_usage_peak_vnf v_memory_usage_mean_vnf v_memory_usage_peak_vnf v_disk_usage_mean_vnf v_disk_usage_peak_vnf. byte_incoming_vnf_int_cp byte_outgoing_vnf_int_cp packet_incoming_vnf_int_cp packet_outgoing_vnf_int_cp	Identifies the performance metric, according to ETSI GS NFV-IFA 027 [7]. Performance metric values shall be set to the corresponding measurement name defined in clause 7.2 of ETSI GS NFV-IFA 027 [7], without appending a sub-counter.
collection_period	no	scalar-unit.time		Describes the recommended periodicity at which to collect the performance information. The VNFM determines if this parameter is considered. The vendor may provide this information as a guidance for creating PmJobs.

6.2.47.3 Definition

The syntax of the VnfcMonitoringParameter data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfcMonitoringParameter:
  derived_from: toasca.datatypes.Root
  description: Represents information on virtualised resource related performance
  metrics applicable to the VNF.
  properties:
    name:
      type: string
      description: Human readable name of the monitoring parameter
      required: true
    performance_metric:
      type: string
      description: Identifies the performance metric, according to ETSI GS NFV-IFA 027.
      required: true
    constraints:
      - valid_values: [ v_cpu_usage_mean_vnf, v_cpu_usage_peak_vnf,
v_memory_usage_mean_vnf, v_memory_usage_peak_vnf, v_disk_usage_mean_vnf,
v_disk_usage_peak_vnf, byte_incoming_vnf_int_cp, byte_outgoing_vnf_int_cp,

```



```

packet_incoming_vnf_int_cp, packet_outgoing_vnf_int_cp ]
collection_period:
  type: scalar-unit.time
  description: Describes the recommended periodicity at
which to collect the performance information.
  required: false
  constraints:
    - greater_than: 0 s

```

6.2.47.4 Examples

None.

6.2.47.5 Additional Requirements

None.

6.2.48 `tosca.datatypes.nfv.VirtualLinkMonitoringParameter`

6.2.48.1 Description

This data type provides information on virtualised resource related performance metrics. Table 6.2.48.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.48.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualLinkMonitoringParameter</code>
Type Qualified Name	<code>toscanfv: VirtualLinkMonitoringParameter</code>
Type URI	<code>tosca.datatypes.nfv.VirtualLinkMonitoringParameter</code>

6.2.48.2 Properties

The properties of the `VirtualLinkMonitoringParameter` data type shall comply with the provisions set out in table 6.2.48.2-1.

Table 6.2.48.2-1: Properties

Name	Required	Type	Constraints	Description
<code>name</code>	no	string		Human readable name of the monitoring parameter.
<code>performance_metric</code>	yes	string	valid values: <code>packet_outgoing_virtual_link</code> <code>packet_incoming_virtual_link</code>	Identifies a performance metric derived from those defined in ETSI GS NFV-IFA 027 [7]. The <code>packet_outgoing_virtual_link</code> and <code>packet_incoming_virtual_link</code> metrics shall be obtained by aggregation the <code>PacketOutgoing</code> and <code>PacketIncoming</code> measurements defined in clause 7.1 of ETSI GS NFV-IFA 027 [7] of all virtual link ports attached to the virtual link to which the metrics apply.
<code>collection_period</code>	no	scalar-unit.time		Describes the recommended periodicity at which to collect the performance information. The VNFM determines if this parameter is considered. The vendor may provide this information as guidance for creating PmJobs.

6.2.48.3 Definition

The syntax of the VirtualLinkMonitoringParameter data type shall comply with the following definition:

```

tosca.datatypes.nfv.VirtualLinkMonitoringParameter:
  derived_from: toska.datatypes.Root
  description: Represents information on virtualised resource related performance
  metrics applicable to the VNF.
  properties:
    name:
      type: string
      description: Human readable name of the monitoring parameter
      required: true
    performance_metric:
      type: string
      description: Identifies a performance metric derived from those defined in ETSI
  GS NFV-IFA 027. The packetOutgoingVirtualLink and packetIncomingVirtualLink metrics
  shall be obtained by aggregation the PacketOutgoing and PacketIncoming measurements
  defined in clause 7.1 of ETSI GS NFV-IFA 027 of all virtual link ports attached to the
  virtual link to which the metrics apply.
      required: true
      constraints:
        - valid_values: [ packet_outgoing_virtual_link, packet_incoming_virtual_link ]
    collection_period:
      type: scalar-unit.time
      description: Describes the recommended periodicity at
  which to collect the performance information.
      required: false
      constraints:
        - greater_than: 0 s

```

6.2.48.4 Examples

None.

6.2.48.5 Additional Requirements

None.

6.2.49 toska.datatypes.nfv.InterfaceDetails

6.2.49.1 Description

The InterfaceDetails data type describes information used to access an interface exposed by a VNF. It corresponds to the interfaceDetails attribute of the VnfInterfaceDetails information element defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.49.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.49.1-1: Type name, shorthand, and URI

Shorthand Name	InterfaceDetails
Type Qualified Name	toscanfv:InterfaceDetails
Type URI	tosca.datatypes.nfv.InterfaceDetails

6.2.49.2 Properties

The properties of the InterfaceDetails data type shall comply with the provisions set out in table 6.2.49.2-1.

Table 6.2.49.2-1: Properties

Name	Required	Type	Constraints	Description
uri_components	no	tosca.datatypes.nfv.UriComponents		Provides components to build a Uniform Resource Identifier (URI) where to access the interface end point.
interface_specific_data	no	map of string		Provides additional details that are specific to the type of interface considered.

6.2.49.3 Definition

The syntax of the InterfaceDetails data type shall comply with the following definition:

```

tosca.datatypes.nfv.InterfaceDetails:
  derived_from: toska.datatypes.Root
  description: information used to access an interface exposed by a VNF
  properties:
    uri_components:
      type: toska.datatypes.nfv.UriComponents
      description: Provides components to build a Uniform Resource Identifier (URI)
where to access the interface end point.
      required: false
    interface_specific_data:
      type: map
      description: Provides additional details that are specific to the type of
interface considered.
      required: false
    entry_schema:
      type: string

```

6.2.49.4 Examples

See clause 6.10.12.

6.2.49.5 Additional Requirements

None.

6.2.50 toska.datatypes.nfv.UriComponents

6.2.50.1 Description

The UriComponents data type describes information used to build a URI that complies with IETF RFC 3986 [8]. Table 6.2.50.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.50.1-1: Type name, shorthand, and URI

Shorthand Name	UriComponents
Type Qualified Name	toscanfv:UriComponents
Type URI	tosca.datatypes.nfv.UriComponents

6.2.50.2 Properties

The properties of the UriComponents data type shall comply with the provisions set out in table 6.2.50.2-1.

Table 6.2.50.2-1: Properties

Name	Required	Type	Constraints	Description
scheme	yes	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the scheme component of a URI, as per IETF RFC 3986 [8]
authority	no	tosca.datatypes.nfv.UriAuthority		Corresponds to the authority component of a URI, as per IETF RFC 3986 [8] See note
path	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the path component of a URI, as per IETF RFC 3986 [8]
query	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the query component of a URI, as per IETF RFC 3986 [8]
fragment	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the fragment component of a URI, as per IETF RFC 3986 [8]
NOTE: If this property is not included while the URI scheme requires it, the VNFM is expected to generate it, based on knowledge of the network configuration of the external CP instance that provides the connectivity for this interface.				

6.2.50.3 Definition

The syntax of the UriComponents data type shall comply with the following definition:

```

tosca.datatypes.nfv.UriComponents:
  derived_from: tosc.datatypes.Root
  description: information used to build a URI that complies with IETF RFC 3986 [8].
  properties:
    scheme:
      type: string # shall comply with IETF RFC 3986
      description: scheme component of a URI.
      required: true
    authority:
      type: tosc.datatypes.nfv.UriAuthority
      description: Authority component of a URI
      required: false
    path:
      type: string # shall comply with IETF RFC 3986
      description: path component of a URI.
      required: false
    query:
      type: string # shall comply with IETF RFC 3986
      description: query component of a URI.
      required: false
    fragment:
      type: string # shall comply with IETF RFC 3986
      description: fragment component of a URI.
      required: false

```

6.2.50.4 Examples

See clause 6.10.12.

6.2.50.5 Additional Requirements

When this datatype is used to provide information for accessing APIs defined in ETSI GS NFV-SOL 002 [i.4], the path property may be included and the query and fragment properties shall be absent. The values of the scheme, authority and path properties form the {apiRoot} of the URI prefix.

6.2.51 toasca.datatypes.nfv.UriAuthority

6.2.51.1 Description

The UriAuthority data type corresponds to the authority component of a URI as specified in IETF RFC 3986 [8]. Table 6.2.51.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.51.1-1: Type name, shorthand, and URI

Shorthand Name	UriAuthority
Type Qualified Name	toscanfv:UriAuthority
Type URI	tosca.datatypes.nfv.UriAuthority

6.2.51.2 Properties

The properties of the UriAuthority data type shall comply with the provisions set out in table 6.2.51.2-1.

Table 6.2.51.2-1: Properties

Name	Required	Type	Constraints	Description
user_info	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the user_info field of the authority component of a URI, as per IETF RFC 3986 [8]. For HTTP and HTTPS URIs, the provisions in sections 2.7.1 and 2.7.2 of IETF RFC 7230 [12] apply, respectively.
host	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the host field of the authority component of a URI, as per IETF RFC 3986 [8]. See note 1.
port	no	string	String values shall comply with IETF RFC 3986 [8]	Corresponds to the port field of the authority component of a URI, as per IETF RFC 3986 [8]. See note 2.
NOTE 1: If this property is not included the VNFM is expected to generate it, based on knowledge of the network configuration of the external CP instance that provides the connectivity for this interface.				
NOTE 2: If this property is not included the default port for the protocol declared by the scheme property of the parent UriComponents structure shall be used unless there are configuration mechanisms applied that are outside the scope of the present document. If no default port exists for the URI scheme, the port property shall be included unless there are configuration mechanisms applied that are outside the scope of the present document.				

6.2.51.3 Definition

The syntax of the UriAuthority data type shall comply with the following definition:

```
tosca.datatypes.nfv.UriAuthority:
  derived_from: toasca.datatypes.Root
  description: information that corresponds to the authority component of a URI as
  specified in IETF RFC 3986 [8]
  properties:
    user_info:
```

```

type: string # shall comply with IETF RFC 3986
description: user_info field of the authority component of a URI
required: false
host:
  type: string # shall comply with IETF RFC 3986
  description: host field of the authority component of a URI
  required: false
port:
  type: string # shall comply with IETF RFC 3986
  description: port field of the authority component of a URI
  required: false

```

6.2.51.4 Examples

See clause 6.10.12.

6.2.51.5 Additional Requirements

When this datatype is used to provide information for accessing APIs defined in ETSI GS NFV-SOL 002 [i.4], the host property and port properties may be included and the user_info property shall not be included. If the host property is included and the value is a registered name, it is assumed that means are in place to resolve the host name to the correct IP address. If the host property is not included, it is assumed that the VNFM will use the IP address associated to one of the connection point instances created from the VnfExpCp and VduCp node types declared as a target of the SupportedVnfInterface policy.

NOTE: This means that if multiple CP instances exist that were created from a particular VnfExtCp or VduCp node template, the VNFM may use any of them to attempt accessing the interface. If no reply is received because the selected CP instance is out of service or is not reachable, the VNFM is expected to try reaching the interface through another CP instance.

6.2.52 toska.datatypes.nfv.VnfProfile

6.2.52.1 Description

The VnfProfile data type describes a profile for instantiating VNFs of a particular NS DF according to a specific VNFD and VNF DF as defined in ETSI GS NFV-IFA 014 [2]. Table 6.2.52.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.52.1-1: Type name, shorthand, and URI

Shorthand Name	VnfProfile
Type Qualified Name	toscanfv:VnfProfile
Type URI	tosca.datatypes.nfv.VnfProfile

6.2.52.2 Properties

The properties of the VnfProfile data type shall comply with the provisions set out in table 6.2.52.2-1.

Table 6.2.52.2-1: Properties

Name	Required	Type	Constraints	Description
instantiation_Level	no	string		Identifier of the instantiation level of the VNF DF to be used for instantiation. If not present, the default instantiation level as declared in the VNFD shall be used.
min_number_of_instances	yes	integer	greater_or_equal : 0	Minimum number of instances of the VNF based on this VNFD that is permitted to exist for this VnfProfile.
max_number_of_instances	yes	integer	greater_or_equal : 0	Maximum number of instances of the VNF based on this VNFD that is permitted to exist for this VnfProfile.

6.2.52.3 Definition

The syntax of the VnfProfile data type shall comply with the following definition:

```

tosca.datatypes.nfv.VnfProfile:
  derived_from: toska.datatypes.Root
  description: describes a profile for instantiating VNFs of a particular NS DF
  according to a specific VNFD and VNF DF.
  properties:
    instantiation_level:
      type: string
      description: Identifier of the instantiation level of the VNF DF to be used for
  instantiation. If not present, the default instantiation level as declared in the VNFD
  shall be used.
      required: false
    min_number_of_instances:
      type: integer
      description: Minimum number of instances of the VNF based on this VNFD that is
  permitted to exist for this VnfProfile.
      required: true
      constraints:
        - greater_or_equal: 0
    max_number_of_instances:
      type: integer
      description: Maximum number of instances of the VNF based on this VNFD that is
  permitted to exist for this VnfProfile.
      required: true
      constraints:
        - greater_or_equal: 0

```

6.2.52.4 Example

None.

6.2.52.5 Additional Requirements

None.

6.3 Artifact Types

6.3.1 `tosca.artifacts.nfv.SwImage`

6.3.1.1 Description

The `SwImage` artifact describes the software image which is directly loaded on the virtualisation container realizing of the VDU or is to be loaded on a virtual storage resource, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.3.1.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.3.1.1-1: Type name, shorthand, and URI

Shorthand Name	<code>SwImage</code>
Type Qualified Name	<code>toscanfv:SwImage</code>
Type URI	<code>tosca.artifacts.nfv.SwImage</code>

6.3.1.2 Description

```
tosca.artifacts.nfv.SwImage:
  derived_from: toasca.artifacts.Deployment.Image
  description: describes the software image which is directly loaded on the
virtualisation container realizing of the VDU or is to be loaded on a virtual storage
resource
```

6.3.2 `tosca.artifacts.Implementation.nfv.Mistral`

6.3.2.1 Description

This artifact type represents a Mistral file that contains Mistral language [i.7] constructs that can be executed within a Mistral workbook. Support of this type is optional.

Table 6.3.2.1-1: Type name, shorthand, and URI

Shorthand Name	<code>Mistral</code>
Type Qualified Name	<code>toscanfv:Mistral</code>
Type URI	<code>tosca.artifacts.Implementation.nfv.Mistral</code>
derived_from	<code>tosca.artifacts.Implementation</code>

6.3.2.2 Definition

The syntax of the Mistral artifact type shall comply with the following definition:

```
tosca.artifacts.Implementation.nfv.Mistral:
  derived_from: toasca.artifacts.Implementation
  description: artifacts for Mistral workflows
  mime_type: application/x-yaml
  file_ext: [ yaml ]
```


6.4 Capability Types

6.4.1 `tosca.capabilities.nfv.VirtualBindable`

6.4.1.1 Description

The `VirtualBindable` capability indicates that the node that includes it can be pointed by a `tosca.relationships.nfv.VirtualBindsTo` relationship type which is used to model the `VduHasCpd` association illustrated in ETSI GS NFV-IFA 011 [1]. Table 6.4.1.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.4.1.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualBindable</code>
Type Qualified Name	<code>toscanfv:VirtualBindable</code>
Type URI	<code>tosca.capabilities.nfv.VirtualBindable</code>

6.4.1.2 Properties

None.

6.4.1.3 Definition

The syntax of the `VirtualBindable` capability type shall comply with the following definition:

```
tosca.capabilities.nfv.VirtualBindable:
  derived_from: toasca.capabilities.Node
  description: Indicates that the node that includes it can be pointed by a
tosca.relationships.nfv.VirtualBindsTo relationship type which is used to model the
VduHasCpd association
```

6.4.2 `tosca.capabilities.nfv.VirtualLinkable`

6.4.2.1 Description

A node type that includes the `VirtualLinkable` capability indicates that it can be pointed by `tosca.relationships.nfv.VirtualLinksTo` relationship type, which is used to model the `VduCpdReferencesVirtualLinkDesc` and the `VnfExtCpdMapsToVirtualLinkDesc` associations illustrated in ETSI GS NFV-IFA 011 [1]. Table 6.4.2.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.4.2.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualLinkable</code>
Type Qualified Name	<code>toscanfv:VirtualLinkable</code>
Type URI	<code>tosca.capabilities.nfv.VirtualLinkable</code>

6.4.2.2 Properties

None.

6.4.2.3 Definition

The syntax of the VirtualCompute capability type shall comply with the following definition:

```
tosca.capabilities.nfv.VirtualLinkable:
  derived_from: tosca.capabilities.Root
  description: A node type that includes the VirtualLinkable capability indicates that
  it can be pointed by tosca.relationships.nfv.VirtualLinksTo relationship type
```

6.4.3 tosca.capabilities.nfv.VirtualCompute

6.4.3.1 Description

The VirtualCompute capability type describes the capabilities related to virtual compute resources, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.4.3.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.4.3.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualCompute
Type Qualified Name	toscanfv:VirtualCompute
Type URI	tosca.capabilities.nfv.VirtualCompute

6.4.3.2 Properties

The properties of the VirtualCompute capability type shall comply with the provisions set out in table 6.4.3.2-1.

Table 6.4.3.2-1: Properties

Name	Required	Type	Constraints	Description
logical_node	no	map of tosca.datatypes.nfv.LogicalNodeData		The Logical Node requirements.
request_additional_capabilities	no	map of tosca.datatypes.nfv.RequestedAdditionalCapability		Describes additional capability for a particular VDU.
compute_requirements	no	map of string		Describes compute requirements.
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.
virtual_local_storage	no	list of tosca.datatypes.nfv.VirtualBlockStorageData		A list of virtual system disks created and destroyed as part of the VM lifecycle.

6.4.3.3 Definition

The syntax of the VirtualCompute capability type shall comply with the following definition:

```
tosca.capabilities.nfv.VirtualCompute:
  derived_from: tosca.capabilities.Node
  description: Describes the capabilities related to virtual compute resources
  properties:
```

```

logical_node:
  type: map
  description: Describes the Logical Node requirements
  required: false
  entry_schema:
    type: toska.datatypes.nfv.LogicalNodeData
requested_additional_capabilities:
  type: map
  description: Describes additional capability for a particular VDU
  required: false
  entry_schema:
    type: toska.datatypes.nfv.RequestedAdditionalCapability
compute_requirements:
  type: map
  required: false
  entry_schema:
    type: string
virtual_memory:
  type: toska.datatypes.nfv.VirtualMemory
  description: Describes virtual memory of the virtualized compute
  required: true
virtual_cpu:
  type: toska.datatypes.nfv.VirtualCpu
  description: Describes virtual CPU(s) of the virtualized compute
  required: true
virtual_local_storage:
  type: list
  description: A list of virtual system disks created and destroyed as part of the
VM lifecycle
  required: false
  entry_schema:
    type: toska.datatypes.nfv.VirtualBlockStorageData
    description: virtual system disk definition

```

6.4.4 toska.capabilities.nfv.VirtualStorage

6.4.4.1 Description

The VirtualStorage capability indicates that the node that includes it can be pointed by a toska.relationships.nfv.AttachesTo relationship type which is used to model the VduHasVirtualStorageDesc association illustrated in ETSI GS NFV-IFA 011 [1]. Table 6.4.4.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.4.4.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualStorage
Type Qualified Name	toscanfv:VirtualStorage
Type URI	tosca.capabilities.nfv.VirtualStorage

6.4.4.2 Definition

The syntax of the VirtualStorage capability type shall comply with the following definition:

```

tosca.capabilities.nfv.VirtualStorage:
  derived_from: toska.capabilities.Root
  description: Describes the attachment capabilities related to Vdu.Storage

```

6.5 Requirements Types

None.

6.6 Relationship Types

6.6.1 `tosca.relationships.nfv.VirtualBindsTo`

6.6.1.1 Description

This relationship type represents an association between `Vdu.Compute` and `VduCp` node types. Table 6.6.1.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.6.1.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualBindsTo</code>
Type Qualified Name	<code>toscanfv:VirtualBindsTo</code>
Type URI	<code>tosca.relationships.nfv.VirtualBindsTo</code>

6.6.1.2 Properties

None.

6.6.1.3 Definition

The syntax of the `VirtualBindsTo` relationship type shall comply with the following definition:

```
tosca.relationships.nfv.VirtualBindsTo:
  derived_from: toasca.relationships.DependsOn
  description: Represents an association relationship between Vdu.Compute and VduCp
node types
  valid_target_types: [ toasca.capabilities.nfv.VirtualBindable ]
```

6.6.2 `tosca.relationships.nfv.VirtualLinksTo`

6.6.2.1 Description

This relationship type represents an association between the `VduCp` and `VnfVirtualLink` node types. Table 6.6.2.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.6.2.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualLinksTo</code>
Type Qualified Name	<code>toscanfv:VirtualLinksTo</code>
Type URI	<code>tosca.relationship.nfv.VirtualLinksTo</code>

6.6.2.2 Properties

None.

6.6.2.3 Definition

The syntax of the VnfVirtualLinksTo relationship type shall comply with the following definition:

```
tosca.relationships.nfv.VirtualLinksTo:
  derived_from: tosca.relationships.DependsOn
  description: Represents an association relationship between the VduCp and
VnfVirtualLink node types
  valid_target_types: [ tosca.capabilities.nfv.VirtualLinkable ]
```

6.6.3 tosca.relationships.nfv.AttachesTo

6.6.3.1 Description

This relationship type represents an association between the Vdu.Compute and one of the following node types: Vdu.VirtualBlockStorage, Vdu.VirtualObjectStorage or Vdu.VirtualFileStorage. Table 6.6.3.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.6.3.1-1: Type name, shorthand, and URI

Shorthand Name	AttachesTo
Type Qualified Name	toscanfv:AttachesTo
Type URI	tosca.relationships.nfv.AttachesTo

6.6.3.2 Properties

None.

6.6.3.3 Definition

The syntax of the AttachesTo relationship type shall comply with the following definition:

```
tosca.relationships.nfv.AttachesTo:
  derived_from: tosca.relationships.Root
  description: Represents an association relationship between the Vdu.Compute and one
of the node types, Vdu.VirtualBlockStorage, Vdu.VirtualObjectStorage or
Vdu.VirtualFileStorage
  valid_target_types: [ tosca.capabilities.nfv.VirtualStorage ]
```

6.7 Interface Types

6.7.1 tosca.interfaces.nfv.Vnflcm

6.7.1.1 Description

The tosca.interfaces.nfv.Vnflcm contains a set of TOSCA operations corresponding to the following VNF LCM operations defined in ETSI GS NFV-IFA 007 [i.1]:

- Instantiate VNF
- Terminate VNF
- Modify VNF information
- Change VNF FlavourChange External VNF Connectivity

- Operate VNF
- Heal VNF
- Scale VNF
- Scale VNF To Level

In addition, the VNFM shall also support TOSCA operations corresponding to preamble and postamble to the execution of the base operation. The name of these operations is constructed according to the following pattern:

- <base_operation_name>_start for a preamble
- <base_operation_name>_end for a postamble

The designations ("_start", "_end") in the name of TOSCA operations are postfixes so that related operations are adjacent in an alphabetical listing.

Table 6.7.1.1-1 specifies the declared names for this interface type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.7.1.1-1: Type name, shorthand, and URI

Shorthand Name	Vnflcm
Type Qualified Name	toscanfv:Vnflcm
Type URI	tosca.interfaces.nfv.Vnflcm

6.7.1.2 Definition

The syntax of the Vnflcm interface type shall comply with the following definition:

```
tosca.interfaces.nfv.Vnflcm:
  derived_from: toasca.interfaces.Root
  description: This interface encompasses a set of TOSCA operations corresponding to
the VNF LCM operations defined in ETSI GS NFV-IFA 007 as well as to preamble and
postamble procedures to the execution of the VNF LCM operations.
  instantiate:
    description: Invoked upon receipt of an Instantiate VNF request
  # inputs:
  # additional_parameters:
  # type: toasca.datatypes.nfv.VnfOperationAdditionalParameters
  # required: false
  # derived types are expected to introduce additional_parameters with its
  # type derived from toasca.datatypes.nfv.VnfOperationAdditionalParameters
  instantiate_start:
    description: Invoked before instantiate
  instantiate_end:
    description: Invoked after instantiate
  terminate:
    description: Invoked upon receipt Terminate VNF request
  # inputs:
  # additional_parameters:
  # type: toasca.datatypes.nfv.VnfOperationAdditionalParameters
  # required: false
  # derived types are expected to introduce additional_parameters with its
  # type derived from toasca.datatypes.nfv.VnfOperationAdditionalParameters
  terminate_start:
    description: Invoked before terminate
  terminate_end:
    description: Invoked after terminate
  modify_information:
    description: Invoked upon receipt of a Modify VNF Information request
```

```

modify_information_start:
  description: Invoked before modify_information
modify_information_end:
  description: Invoked after modify_information
change_flavour:
  description: Invoked upon receipt of a Change VNF Flavour request
  # inputs:
  # additional_parameters:
  #   # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  #   # required: false
  #   # derived types are expected to introduce additional_parameters with its
  #   # type derived from tosca.datatypes.nfv.VnfOperationAdditionalParameters
change_flavour_start:
  description: Invoked before change_flavour
change_flavour_end:
  description: Invoked after change_flavour
change_external_connectivity:
  description: Invoked upon receipt of a Change External VNF Connectivity request
  # inputs:
  # additional_parameters:
  #   # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  #   # required: false
  #   # derived types are expected to introduce additional_parameters with its
  #   # type derived from tosca.datatypes.nfv.VnfOperationAdditionalParameters
change_external_connectivity_start:
  description: Invoked before change_external_connectivity
change_external_connectivity_end:
  description: Invoked after change_external_connectivity
operate:
  description: Invoked upon receipt of an Operate VNF request
  # inputs:
  # additional_parameters:
  #   # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  #   # required: false
  #   # derived types are expected to introduce additional_parameters with its
  #   # type derived from tosca.datatypes.nfv.VnfOperationAdditionalParameters
operate_start:
  description: Invoked before operate
operate_end:
  description: Invoked after operate
heal:
  description: Invoked upon receipt of a Heal VNF request
  # inputs:
  # additional_parameters:
  #   # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  #   # required: false
  #   # derived types are expected to introduce additional_parameters with its
  #   # type derived from tosca.datatypes.nfv.VnfOperationAdditionalParameters
heal_start:
  description: Invoked before heal
heal_end:
  description: Invoked after heal
scale:
  description: Invoked upon receipt of a Scale VNF request
  # inputs:
  # additional_parameters:
  #   # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  #   # required: false
  #   # derived types are expected to introduce additional_parameters with its
  #   # type derived from tosca.datatypes.nfv.VnfOperationAdditionalParameters
scale_start:

```

```

description: Invoked before scale
scale_end:
description: Invoked after scale
scale_to_level:
description: Invoked upon receipt of a Scale VNF to Level request
# inputs:
# additional_parameters:
# type: toasca.datatypes.nfv.VnfOperationAdditionalParameters
# required: false
# derived types are expected to introduce additional_parameters with its
# type derived from toasca.datatypes.nfv.VnfOperationAdditionalParameters
scale_to_level_start:
description: Invoked before scale_to_level
scale_to_level_end:
description: Invoked after scale_to_level

```

6.7.1.3 Additional Requirements

All VNF supported LCM operations shall be listed in the service template, except "instantiate" and "terminate" that may be omitted, as specified in ETSI GS NFV-IFA 011 [1] for the supportedOperation attribute of a deployment flavour.

The implementation and inputs keynames specified in TOSCA-Simple-Profile-YAML-v1.2 [3] for an operation definition may be included for each operation listed in the Vnflcm interface definition.

If a TOSCA operation representing a VNF LCM operation is listed in the service template without an associated implementation, then it means that:

- the VNF LCM operation is supported (i.e. this is the manifestation of the supportedOperation attribute as per ETSI GS NFV-IFA 011 [1]); and
- the processing logic associated with the LCM operation is the default implementation provided by the VNFM.

If an implementation is associated to a TOSCA operation that represents a preamble or a postamble to a VNF LCM operation, the implementation logic is executed before or after the execution of the VNF LCM operation implementation, respectively.

The VNFM shall make available all parameters from the message invoking the VNF LCM operation as inputs to the corresponding TOSCA interface operations.

In the operation definitions on the Vnflcm interface, the additional_parameters (VNF-specific extension of the toasca.datatypes.nfv.VnfOperationAdditionalParameters) of the inputs section describes the name and type of the additional parameters (additionalParams) that can be submitted in the VNF LCM operation request. See an example in clause 6.2.37 (tosca.datatypes.nfv.VnfOperationAdditionalParameters).

The inputs keyname can also be used to specify additional input parameters for executing the TOSCA operation, beyond those received in the VNF LCM operation request. To distinguish them from the latter ones, such input parameters shall not be named "additional_parameters".

The implementation of the operation corresponding to preamble and postamble TOSCA operations (instantiate_start, instantiate_end, scale_start, scale_end, etc.), if present, shall be invoked with the same parameters as the corresponding base operations ones (instantiate, scale, etc.). The inputs of the operations corresponding to the postamble and preamble operations shall not be defined in the VNFD.

6.7.1.4 Support of LCM scripts

In ETSI GS NFV-IFA 011 [1], the definition of the "LifeCycleManagementScript" information element of the VNFD associates scripts with events, where an event can be an external or an internal stimulus. These events are mapped to TOSCA operations of the VNF node type in the following way:

- external stimuli are mapped to TOSCA operations corresponding to the VNF LCM operations defined in ETSI GS NFV-IFA 007 [i.1];

- internal stimuli are mapped to preamble and postamble of these TOSCA operations;
- events that cannot be mapped to these TOSCA operations (lcmTransitionEvent as described in ETSI GS NFV-IFA 011 [1], clause 7.1.13) can be mapped to further TOSCA operations by extending the TOSCA interface.

The LCM scripts can be regarded as artifacts that provide a VNF-specific implementation of the TOSCA operation corresponding to the stimulus. The script input parameters shall be provided to the script according to the declaration in the inputs field of the operation definition. The artifact type definition shall enable identifying the DSL used by the script. The artifact type definition for Python is provided in section 5.4.4.1 of TOSCA-Simple-Profile-YAML-v1.2 [3]. The artifact definition for Mistral is provided in clause A.7.2 of the present document.

NOTE: As all input parameters needed for operations corresponding to external and internal stimuli are defined in the "input parameters of the external stimuli operations", the VNF Designer is expected to make the list of parameters as complete as needed to handle not only the external stimuli but also the internal stimuli.

6.7.1.5 Examples

The following example template fragments illustrate the concepts.

```
tosca_definitions_version: tosca_simple_yaml_1_2

imports:
- ..

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    ..

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    ..

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    ..
  interfaces:
    Vnflcm:
      instantiate:
      ..
```

In the above example, as there is no implementation and inputs specified to the operations, the built-in implementation of the operation is invoked when the Instantiate VNF request is received on the LCM interface of the Or-Vnfm reference point. The received parameters (flavourId, instantiationLevelId, ...) are passed to the built-in implementation (as flavour_id, instantiation_level_id).

```
tosca_definitions_version: tosca_simple_yaml_1_2

imports:
- ..

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    ..

topology_template:
```

```

substitution_mappings:
  node_type: MyCompany.SunshineDB.1_0.1_0
  ..

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    ..
  interfaces:
    Vnflcm:
      instantiate:
        implementation: instantiate-script
      ..

```

In the above example, the `instantiate-script` is invoked when the `Instantiate VNF` request is received, passing the received parameters to it similarly to the previous example. This example does not imply a one-to-one mapping between operations and script names.

```

tosca_definitions_version: tosca_simple_yaml_1_2

imports:
- ..

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    ..

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    ..

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    ..
  interfaces:
    Vnflcm:
      ..
      scale_start: pre-scale-script
      scale_end: post-scale-script
      ..

```

In the above example, LCM scripts are associated with the "scale start" and "scale end" internal stimuli. As no script is associated to the `scale` operation, its default implementation runs (after running the `pre-scale-script`, and before running the `post-scale-script`).

```

tosca_definitions_version: tosca_simple_yaml_1_2

imports:
- ..

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    ..

```

```

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    ..

  node_templates:
    SunshineDB:
      type: MyCompany.SunshineDB.1_0.1_0
      ..
    interfaces:
      Vnflcm:
        instantiate:
          implementation: instantiate-script
          inputs:
            script_input_1: value_1
            script_input_2: value_2

  artifacts:
    instantiate-script:
      description: Instantiate workflow script
      type: toasca.artifacts.Implementation.Python
      file: instantiate.py
      #repository: ..
      #deploy_path: ..
      ..

```

In the above example:

- The inputs section provides additional input values to the instantiate-script (i.e. the manifestation of the scriptInput attribute of LifecycleManagementScript as defined in ETSI GS NFV-IFA 011 [1]).

NOTE: There is another kind of input called `additional_parameters` dedicated to the additional parameters (additionalParams) received in the message invoking the VNF LCM operation; this input is not illustrated by the above examples; see clause 6.2.43.4 on how to declare `additional_parameters` in the derived VNF node type.

- TOSCA artifacts definition is used to convey the type of DSL used as a scripting language that is associated with an operation (i.e. the manifestation of the scriptDsl attribute of LifecycleManagementScript as per ETSI GS NFV-IFA 011 [1]).

6.8 Node Types

6.8.1 toasca.nodes.nfv.VNF

6.8.1.1 Description

The VNF node type is the generic abstract type from which all VNF specific abstract node types shall be derived to form, together with other node types, the TOSCA service template(s) representing the VNFD information element as defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.1.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.1.1-1: Type name, shorthand, and URI

Shorthand Name	VNF
Type Qualified Name	toscanfv:VNF
Type URI	tosca.nodes.nfv.VNF

6.8.1.2 Properties

The properties of the VNF node type shall comply with the provisions set out in table 6.8.1.2-1.

Table 6.8.1.2-1: Properties

Name	Required	Type	Constraints	Description
descriptor_id	yes	string		Identifier of this VNFD information element. This attribute shall be globally unique. The VNFD Identifier shall be used as the unique identifier of the VNF Package that contains this VNFD. Any modification of the content of the VNFD or the VNF Package shall result in a new VNFD Identifier.
descriptor_version	yes	string		Identifies the version of the VNFD.
provider	yes	string		Provider of the VNF and of the VNFD.
product_name	yes	string		Name to identify the VNF Product. Invariant for the VNF Product lifetime.
software_version	yes	string		Software version of the VNF. This is changed when there is any change to the software that is included in the VNF Package.
product_info_name	no	string		Human readable name for the VNF Product. Can change during the VNF Product lifetime.
product_info_description	no	string		Human readable description of the VNF Product. Can change during the VNF Product lifetime.
vnfm_info	yes	list of string		Identifies VNFM(s) compatible with the VNF described in this version of the VNFD. To indicate that a VNF can be managed by any ETSI NFV compliant VNFM, the string value shall be the concatenation of the string "etsivnfm" and the minimum version of ETSI GS NFV-SOL 002 [i.4] to be supported by this VNFM (e.g. etsivnfm:v2.3.1). If the VNF is compatible with multiple versions, multiple values may be included. See note 1. To indicate a specific VNFM product, the string value shall be the concatenation of the IANA enterprise number of the VNFM provider [5], followed by a product-specific string.

Name	Required	Type	Constraints	Description
localization_languages	no	list of string	Valid values: string values that comply with IETF RFC 5646 [13]	Information about localization languages of the VNF (includes e.g. strings in the VNFD). This allows to provide one or more localization languages to support selecting a specific localization language at VNF instantiation time.
default_localization_language	no	string	Valid values: string values that comply with IETF RFC 5646 [13]	Default localization language that is instantiated if no information about selected localization language is available. Shall be present if "localizationLanguage" is present and shall be absent otherwise.
configurable_properties	no	tosca.datatype s.nfv.VnfConfigurableProperties		Describes the configurable properties of the VNF (e.g. related to auto scaling and auto healing).
modifiable_attributes	no	tosca.datatype s.nfv.VnfInfoModifiableAttributes		Describes the modifiable attributes of the VNF.
lcm_operations_configuration	no	tosca.datatype s.nfv.VnfLcmOperationsConfiguration		Describes the configuration parameters for the VNF LCM operations.
monitoring_parameters	no	list of tosca.datatype s.nfv.VnfMonitoringParameter		Describes monitoring parameters applicable to the VNF.
flavour_id	yes	string		Identifier of this DF within the VNFD.
flavour_description	yes	string		Human readable description of the DF.
vnf_profile	no	tosca.datatype s.nfv.VnfProfile		Describes a profile for instantiating VNFs of a particular NS DF according to a specific VNFD and VNF DF. See note 2.
NOTE 1: When LCM scripts are used, the support of this minimum version might not be sufficient to ensure that the VNF can be managed by a VNFM. The support of the domain specific language(s) used by these LCM scripts is another criterion for determining the compatibility of the VNF with a VNFM.				
NOTE 2: This property is only used in a NSD service template when describing a VNF node template with the corresponding VnfProfile information.				

The value of the descriptor_id string shall comply with an UUID format as specified in section 3 of [9].

The syntax of the `vnfm_info` string values shall comply with the following ABNF [6] snippet:

```
value = any_etsi_nfv_compliant_product | product_specific
any_etsi_nfv_compliant_product = "etsivnfm" SEP "version"

version = "v" version_identifier
version_identifier = 1*2DIGIT DOT 1*2DIGIT DOT 1*2DIGIT
; the version identifier is encoded as a sequence of 1 or 2 digits separated by dots representing
the 3 fields (major, technical and editorial) of the version of an ETSI deliverable.

product_specific = enterprise_number SEP product_specific_string
enterprise_number = 1*DIGIT
product_specific_string = *(ALPHA / DIGIT / "-" / ".")

SEP = ":"
DOT = "."
```

6.8.1.3 Attributes

None.

6.8.1.4 Requirements

The requirements of the VNF node type shall comply with the provisions set out in table 6.8.1.4-1.

Table 6.8.1.4-1: Requirements

Name	Required	Type	Constraints	Description
virtual_link	yes	tosca.capabilities.nfv.VirtualLinkable		Describes the requirements for linking to virtual link

6.8.1.5 Capabilities

None.

6.8.1.6 Definition

The syntax of the VNF node type shall comply with the following definition:

```
tosca.nodes.nfv.VNF:
  derived_from: tosca.nodes.Root
  description: The generic abstract type from which all VNF specific abstract node
types shall be derived to form, together with other node types, the TOSCA service
template(s) representing the VNFD
  properties:
    descriptor_id: # instead of vnfd_id
      type: string # GUID
      description: Globally unique identifier of the VNFD
      required: true
    descriptor_version: # instead of vnfd_version
      type: string
      description: Identifies the version of the VNFD
      required: true
    provider: # instead of vnf_provider
      type: string
      description: Provider of the VNF and of the VNFD
      required: true
    product_name: # instead of vnf_product_name
      type: string
      description: Human readable name for the VNF Product
      required: true
```

```

software_version: # instead of vnf_software_version
  type: string
  description: Software version of the VNF
  required: true
product_info_name: # instead of vnf_product_info_name
  type: string
  description: Human readable name for the VNF Product
  required: false
product_info_description: # instead of vnf_product_info_description
  type: string
  description: Human readable description of the VNF Product
  required: false
vnfm_info:
  type: list
  required: true
  description: Identifies VNFM(s) compatible with the VNF
  entry_schema:
    type: string
localization_languages:
  type: list
  description: Information about localization languages of the VNF
  required: false
  entry_schema:
    type: string #IETF RFC 5646 string
default_localization_language:
  type: string #IETF RFC 5646 string
  description: Default localization language that is instantiated if no information
about selected localization language is available
  required: false
#configurable_properties:
#type: toasca.datatypes.nfv.VnfConfigurableProperties
#description: Describes the configurable properties of the VNF
#required: false
# derived types are expected to introduce configurable_properties
# with its type derived from toasca.datatypes.nfv.VnfConfigurableProperties
#modifiable_attributes:
#type: toasca.datatypes.nfv.VnfInfoModifiableAttributes
#description: Describes the modifiable attributes of the VNF
#required: false
# derived types are expected to introduce modifiable_attributes
# with its type derived from
# toasca.datatypes.nfv.VnfInfoModifiableAttributes
lcm_operations_configuration:
  type: toasca.datatypes.nfv.VnfLcmOperationsConfiguration
  description: Describes the configuration parameters for the VNF LCM operations
  required: false
monitoring_parameters:
  type: list
  entry_schema:
    type: toasca.datatypes.nfv.VnfMonitoringParameter
  description: Describes monitoring parameters applicable to the VNF.
  required: false
flavour_id:
  type: string
  description: Identifier of the Deployment Flavour within the VNFD
  required: true
flavour_description:
  type: string
  description: Human readable description of the DF
  required: true
vnf_profile:

```

```

type: toska.datatypes.nfv.VnfProfile
description: Describes a profile for instantiating VNFs of a particular NS DF
according to a specific VNFD and VNF DF
required: false
requirements:
- virtual_link:
  capability: toska.capabilities.nfv.VirtualLinkable
  relationship: toska.relationships.nfv.VirtualLinksTo
  occurrences: [ 0, 1 ]
# Additional requirements shall be defined in the VNF specific node type (deriving
from toska.nodes.nfv.VNF) corresponding to NS virtual links that need to connect to
VnfExtCps
interfaces:
  Vnflcm:
    type: toska.interfaces.nfv.Vnflcm

```

6.8.1.7 Artifact

None.

6.8.1.8 Additional Requirements

For a given VNFD, a new VNF node type shall be defined following the below requirements:

- a) The node type shall be derived from: toska.nodes.nfv.VNF.
- b) All properties listed in toska.nodes.nfv.VNF where the "required:" field is set to "true" shall be included with their values indicated as constraints and as default values.

NOTE 1: Indicating their values as default allows not to include them in property assignments in node templates, e.g. in the NSD.

- c) An empty string shall be indicated as the default value of the flavour_description property.
- d) The capabilities, requirements, interfaces of toska.nodes.nfv.VNF shall be preserved.
- e) Depending on the number of external connection points of the VNF that need to connect to NS virtual links, additional requirements for VirtualLinkable capability shall be defined. In this case, it is the VNFD author's choice to use the requirement for VirtualLinkable capability defined in the toska.nodes.nfv.VNF node type or use only the additional requirements defined in the derived VNF specific node type.
- f) The rule for naming this node type in the service template should be:
 - provider.product_name.software_version.descriptor_version, by concatenating the values of the corresponding properties of the created VNF node type.

NOTE 2: If the software_version value or descriptor_version value contains a dot (i.e. "."), this character should be replaced with an underscore (i.e. "_").

VNF Providers shall use the following types to derive the VNF specific modifiable attributes and additional configurable properties:

- toska.datatypes.nfv.VnfInfoModifiableAttributesExtensions.
- toska.datatypes.nfv.VnfInfoModifiableAttributesMetadata.
- toska.datatypes.nfv.VnfAdditionalConfigurableProperties.
- toska.datatypes.nfv.VnfInfoModifiableAttributes.
- toska.datatypes.nfv.VnfConfigurableProperties.

See illustrative examples in clauses 6.8.1.9, 6.2.35.4 and 6.2.31.4.

Node templates of the VNF specific node type shall not include the `vnf_profile` property when they are part of a VNFD service template.

For a given NSD, when describing a referenced VNFD as a node templates, the `vnf_profile` property shall be included with a valid value.

6.8.1.9 Example

Example usage of `modifiable_attributes`:

```
tosca_definitions_version: toska_simple_yaml_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: toska.nodes.nfv.VNF
    properties:
      flavour_id:
        constraints:
          - valid_values: [ simple, complex ]
    modifiable_attributes:
      type: mycompany.datatypes.nfv.VnfInfoModifiableAttributes

data_types:
  mycompany.datatypes.nfv.VnfInfoModifiableAttributes:
    derived_from: toska.datatypes.nfv.VnfInfoModifiableAttributes
    properties:
      extensions:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions

  mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions:
    derived_from: toska.datatypes.nfv.VnfInfoModifiableAttributesExtensions
    properties:
      http_proxy:
        type: string
        required: true
      https_proxy:
        type: string
        required: false
```

Example usage of `lcm_operations_configuration`:

Top level service template:

```
tosca_definitions_version: toska_simple_yaml_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: toska.nodes.nfv.VNF
    properties:
      flavour_id:
        type: string
        constraints:
          - valid_values: [ simple, complex ]
    lcm_operations_configuration:
      scale:
        scaling_by_more_than_one_step_supported: true
      scale_to_level:
        arbitrary_target_levels_supported: true
      heal:
```

```

causes:
  - service_unavailable
  - performance_degraded
terminate:
  min_graceful_termination_timeout: 60 s
  max_recommended_graceful_termination_timeout: 600 s
operate:
  min_graceful_stop_timeout: 60 s
  max_recommended_graceful_stop_timeout: 600 s

```

Example usage of describing a VNF node template with `vnf_profile` in a NSD TOSCA service template:

```

tosca_definitions_version: tosca_simple_yaml_1_2

description: an example of NSD TOSCA service template

topology template:
...

node_templates:
  VNF_1:
    type: tosca.nodes.nfv.exampleVNF
    properties:
      flavour_id: small
      descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      provider: MyCompany
      product_name: SunshineDB
      software_version: 1.0
      descriptor_version: 1.0
      vnf_profile:
        instantiation_level: level_1
        min_number_of_instances: 2
        max_number_of_instances: 6
    # other properties omitted for brevity
    requirements:
      - virtual_link: NsVirtualLink

```

6.8.2 tosca.nodes.nfv.VnfExtCp

6.8.2.1 Description

The `VnfExtCp` node type represents the `VnfExtCpd` information element as defined in ETSI GS NFV-IFA 011 [1], which describes a logical external connection point, exposed by this VNF enabling connecting with an external Virtual Link. Table 6.8.2.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.2.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VnfExtCp</code>
Type Qualified Name	<code>tosca.nfv.VnfExtCp</code>
Type URI	<code>tosca.nodes.nfv.VnfExtCp</code>

6.8.2.2 Properties

The properties of the `VnfExtCp` node type shall comply with the provisions set out in table 6.8.2.2-1.

Table 6.8.2.2-1: Properties

Name	Required	Type	Constraints	Description
virtual_network_interface_requirements	no	list of toska.datatypes.nfv.VirtualNetworkInterfaceRequirements		The actual virtual NIC requirements that is been assigned when instantiating the connection point.

6.8.2.3 Attributes

None.

6.8.2.4 Requirements

The requirements of the VnfExtCp node type shall comply with the provisions set out in table 6.8.2.4-1.

Table 6.8.2.4-1: Requirements

Name	Required	Type	Constraints	Description
external_virtual_link	no	tosca.capabilities.nfv.VirtualLinkable		Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability
internal_virtual_link	yes	tosca.capabilities.nfv.VirtualLinkable		Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability

6.8.2.5 Definition

The syntax of the VnfExtCp node type shall comply with the following definition:

```
tosca.nodes.nfv.VnfExtCp:
  derived_from: toska.nodes.nfv.Cp
  description: Describes a logical external connection point, exposed by the VNF
  enabling connection with an external Virtual Link
  properties:
    virtual_network_interface_requirements:
      type: list
      description: The actual virtual NIC requirements that is been assigned when
  instantiating the connection point
      required: false
      entry_schema:
        type: toska.datatypes.nfv.VirtualNetworkInterfaceRequirements
  requirements:
    - external_virtual_link:
      capability: toska.capabilities.nfv.VirtualLinkable
      relationship: toska.relationships.nfv.VirtualLinksTo
    - internal_virtual_link: #name in ETSI NFV IFA 011 v0.7.3: intVirtualLinkDesc
      capability: toska.capabilities.nfv.VirtualLinkable
      relationship: toska.relationships.nfv.VirtualLinksTo
```

6.8.2.6 Additional Requirements

A node template of this type is used to represent a VNF external connection point only in the case the VnfExtCp is connected to an internal virtual link. The node template has the following requirements:

- internal_virtual_link requirement to allow to connect it to an internal virtual link;
- external_virtual_link requirement to allow to connect it to an external virtual link.

In the case where a VNF external connection point is re-exposing a VduCp (internal connection point), the VduCp node type shall be used in the service template, instead of the VnfExtCp node type.

6.8.2.7 Example

In a typical scenario, the VnfExtCp node template will be part of a service template representing a certain VNF deployment flavour. The service template substitutes for an abstract VNF node type. In this substitution, the virtual_link requirement of the abstract VNF node type is mapped to the external_virtual_link requirement of the VnfExtCp node. This example is illustrated in clause A 3.2.

When a VNF external connection point re-exposes a Vdu connection point, the service template does not require an explicit node template of type VnfExtCp in a typical scenario where an abstract VNF node type is substituted by a service template representing a certain VNF deployment flavour. In this substitution, the virtual_link requirement of the abstract VNF node type is mapped to the virtual_link requirement of the VduCp node. This example is illustrated in clause A 3.1.

6.8.3 toska.nodes.nfv.Vdu.Compute

6.8.3.1 Description

The Vdu.Compute node type describes the virtual compute part of a VDU which is a construct supporting the description of the deployment and operational behaviour of a VNFC, as defined in ETSI GS NFV-IFA 011 [1].

Table 6.8.3.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.3.1-1: Type name, shorthand, and URI

Shorthand Name	Vdu.Compute
Type Qualified Name	toscanfv:Vdu.Compute
Type URI	tosca.nodes.nfv.Vdu.Compute

6.8.3.2 Properties

The properties of the Vdu.Compute node type shall comply with the provisions set out in table 6.8.3.2-1.

Table 6.8.3.2-1: 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 node template name from which a valid boot device is created e.g. VirtualBlockStorage node template from which a VirtualBlockStorage 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.
monitoring_parameters	no	list of toska.datatypes.nfv.VnfcMonitoringParameter		Describes monitoring parameters applicable to a VNFC based on this VDU.

Name	Required	Type	Constraints	Description
configurable_properties	no	map of tosca.datatypes.nfv.VnfcConfigurableProperties		Describes the configurable properties of all VNFC instances based on this VDU.
boot_data	no	string		Contains a string or a URL to a file contained in the VNF package used to customize a virtualised compute resource at boot time. The bootData may contain variable parts that are replaced by deployment specific values before being sent to the VIM. For the parameters with "volatile" values, the parameters of each variable part shall be declared in the tosca.datatypes.nfv.VnfOperationAdditionalParameters; for parameters with "persistent" values during the lifetime of the VNF instance, the parameters shall be declared in the tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions.
vdu_profile	yes	tosca.datatypes.nfv.VduProfile		Defines additional instantiation data for the VDU.Compute node.
sw_image_data	no	tosca.datatypes.nfv.SwImageData		Defines information related to a SwImage artifact used by this Vdu.Compute node.

6.8.3.3 Attributes

None.

6.8.3.4 Requirements

The requirements of the Vdu.Compute node type shall comply with the provisions set out in table 6.8.3.4-1.

Table 6.8.3.4-1: Requirements

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

6.8.3.5 Capabilities

The capabilities of the Vdu.Compute node type shall comply with the provisions set out in table 6.8.3.5-1.

Table 6.8.3.5-1: Capabilities

Name	Type	Constraints	Description
virtual_compute	tosca.capabilities.nfv.VirtualCompute		Describes virtual compute resources capabilities.
virtual_binding	tosca.capabilities.nfv.VirtualBindable		Defines ability of VirtualBindable. See note.
NOTE: Due to current grammar limitations in TOSCA-Simple-Profile-YAML-v1.2 [3], the Vdu.Compute node template cannot limit the number of the virtual_binding capability occurrences. It is the VNFD author's responsibility to make sure that the number of VduCps connected to a particular Vdu.Compute node is compatible with the virtualised resources assigned to this node.			

6.8.3.6 Definition

The syntax of the Vdu.Compute node type shall comply with the following definition:

```

tosca.nodes.nfv.Vdu.Compute:
  derived_from: toska.nodes.Root
  description: Describes the virtual compute part of a VDU which is a construct
  supporting the description of the deployment and operational behavior of a VNFC
  properties:
    name:
      type: string
      description: Human readable name of the VDU
      required: true
    description:
      type: string
      description: Human readable description of the VDU
      required: true
    boot_order:
      type: list # explicit index (boot index) not necessary, contrary to IFA011
      description: References a node template name from which a valid boot device is
  created
      required: false
    entry_schema:
      type: string
    nfvi_constraints:
      type: list
      description: Describes constraints on the NFVI for the VNFC instance(s) created
  from this VDU
      required: false
    entry_schema:
      type: string
    monitoring_parameters:
      type: list
      description: Describes monitoring parameters applicable to a VNFC instantiated
  from this VDU
      required: false
    entry_schema:
      type: toska.datatypes.nfv.VnfcMonitoringParameter
  #configurable_properties:
    #type: toska.datatypes.nfv.VnfcConfigurableProperties
    #required: false
    # derived types are expected to introduce
    # configurable_properties with its type derived from
    # toska.datatypes.nfv.VnfcConfigurableProperties
  vdu_profile:
    type: toska.datatypes.nfv.VduProfile
    description: Defines additional instantiation data for the VDU.Compute node
    required: true
  sw_image_data:
    type: toska.datatypes.nfv.SwImageData
    description: Defines information related to a SwImage artifact used by this
  Vdu.Compute node
    required: false # property is required when the node template has an associated
  artifact of type toska.artifacts.nfv.SwImage and not required otherwise
  boot_data:
    type: string
    description: Contains a string or a URL to a file contained in the VNF package
  used to customize a virtualised compute resource at boot time. The bootData may
  contain variable parts that are replaced by deployment specific values before being
  sent to the VIM.
    required: false

```

```

capabilities:
  virtual_compute:
    type: tosca.capabilities.nfv.VirtualCompute
    occurrences: [ 1, 1 ]
  virtual_binding:
    type: tosca.capabilities.nfv.VirtualBindable
    occurrences: [ 1, UNBOUNDED ]
requirements:
  - virtual_storage:
    capability: tosca.capabilities.nfv.VirtualStorage
    relationship: tosca.relationships.nfv.AttachesTo
    occurrences: [ 0, UNBOUNDED ]

```

6.8.3.7 Additional requirements

Node templates of type `tosca.nodes.nfv.Vdu.Compute` may contain an artifact definition of type `tosca.artifacts.nfv.SwImage`.

VNF Providers shall use the datatype `tosca.datatypes.nfv.VnfcConfigurableProperties` to derive the type of `configurable_properties` when introducing this property with VDU-specific elements in a `Vdu.Compute` specific node type.

The node template name of type `tosca.nodes.nfv.Vdu.Compute` fulfills the purpose of the 'virtualComputeDescId' attribute of the `virtualComputeDesc` information element in ETSI GS NFV-IFA 011 [1] and hence it will be used in APIs to identify the virtual compute id (`vnfdVirtualComputeDescId`).

NOTE: The use of the node template name of type `tosca.nodes.nfv.Vdu.Compute` for the 'virtualComputeDescId' attribute of the `virtualComputeDesc` information element in ETSI GS NFV-IFA 011 [1] implies in the present document version a one-to-one mapping of `virtualComputeDesc` with VDU. This deviates from the ETSI GS NFV-IFA 011 [1] modeling that defines a mapping where a `virtualComputeDesc` can be reused by one or more VDU, i.e., it implies a one-to-many mapping of `virtualComputeDesc` with VDU. This can have an impact in the determination of the number of compute flavours needed to be created with the VIM.

6.8.3.8 Example

This example illustrates boot data values as string.

```

tosca_definitions_version: tosca_simple_yaml_1_2
..
node_types:
  mycompany.nodes.nfv.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ..
      modifiable_attributes:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributes
      ..
data_types:
  mycompany.datatypes.nfv.VnfInfoModifiableAttributes:
    derived_from: tosca.datatypes.nfv.VnfInfoModifiableAttributes
    properties:
      extensions:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions
      required: false
  mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions:
    derived_from: tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions
    properties:

```

```

    http_proxy:
      type: string
      required: true
    https_proxy:
      type: string
      required: false

topology_template:
  inputs:
    modifiable_attributes:
      type: mycompany.datatypes.nfv.VnfInfoModifiableAttributes

substitution_mappings:
  node_type: mycompany.nodes.nfv.SunshineDB.1_0.1_0
  ..

node_templates:
  vnf:
    type: mycompany.nodes.nfv.SunshineDB.1_0.1_0
    properties:
      ..
      modifiable_attributes: { get_input: modifiable_attributes }

  dbBackend:
    type: toska.nodes.nfv.Vdu.Compute
    properties:
      ..
      boot_data: { concat: [
        "#!/bin/bash\n",
        "echo setting HTTP proxy to: ", { get_property: [vnf, modifiable_attributes,
extensions, http_proxy ] }, "\n",
        "..."]
      ] }
      ..

```

This example illustrates fetching the boot data value from a file.

```

tosca_definitions_version: toska_simple_yaml_1_2
..
topology_template:
  ..

node_templates:
  dbBackend:
    type: toska.nodes.nfv.Vdu.Compute
    properties:
      ..
      boot_data: { get_artifact: [ SELF, user_data ] }
      ..
  artifacts:
    user_data:
      file: db-backend-init.yaml
      type: toska.artifacts.File
    ..

```


6.8.4 toska.nodes.nfv.Vdu.VirtualBlockStorage

6.8.4.1 Description

The VirtualBlockStorage node type describes the specifications of requirements related to virtual block storage resources, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.4.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.4.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualBlockStorage
Type Qualified Name	toscanfv:VirtualBlockStorage
Type URI	tosca.nodes.nfv.Vdu.VirtualBlockStorage

6.8.4.2 Properties

The properties of the VirtualBlockStorage node type shall comply with the provisions set out in table 6.8.4.2-1.

Table 6.8.4.2-1: Properties

Name	Required	Type	Constraints	Description
virtual_block_storage_data	yes	tosca.datatypes.nfv.VirtualBlockStorage		Describes the block storage characteristics.
sw_image_data	no	tosca.datatypes.nfv.SwImageData		Defines information related to a SwImage artifact used by this Vdu.Compute node.

6.8.4.3 Attributes

None

6.8.4.4 Requirements

None.

6.8.4.5 Capabilities

The capabilities of the VirtualBlockStorage node type shall comply with the provisions set out in table 6.8.4.5-1.

Table 6.8.4.5-1: Capabilities

Name	Type	Constraints	Description
virtual_storage	tosca.capabilities.nfv.VirtualStorage		Defines the capabilities of virtual_storage.

6.8.4.6 Definition

The syntax of the VirtualBlockStorage node type shall comply with the following definition:

```
tosca.nodes.nfv.Vdu.VirtualBlockStorage:
  derived_from: toska.nodes.Root
  description: This node type describes the specifications of requirements related to
  virtual block storage resources
  properties:
    virtual_block_storage_data:
      type: toska.datatypes.nfv.VirtualBlockStorageData
      description: Describes the block storage characteristics.
```

```

required: true
sw_image_data:
  type: toska.datatypes.nfv.SwImageData
  description: Defines information related to a SwImage artifact used by this
Vdu.Compute node.
  required: false # property is required when the node template has an associated
artifact of type toska.artifacts.nfv.SwImage and not required otherwise
capabilities:
  virtual_storage:
    type: toska.capabilities.nfv.VirtualStorage
    description: Defines the capabilities of virtual_storage.

```

6.8.4.7 Additional requirements

Node templates of type `tosca.nodes.nfv.Vdu.VirtualBlockStorage` may contain an artifact definition of type `tosca.artifacts.nfv.SwImage`.

6.8.5 `tosca.nodes.nfv.Vdu.VirtualObjectStorage`

6.8.5.1 Description

The `VirtualObjectStorage` node type describes the specifications of requirements related to virtual object storage resources, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.5.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.5.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VirtualObjectStorage</code>
Type Qualified Name	<code>toscanfv:VirtualObjectStorage</code>
Type URI	<code>tosca.nodes.nfv.Vdu.VirtualObjectStorage</code>

6.8.5.2 Properties

The properties of the `VirtualObjectStorage` node type shall comply with the provisions set out in table 6.8.5.2-1.

Table 6.8.5.2-1: Properties

Name	Required	Type	Constraints	Description
<code>virtual_object_storage_data</code>	yes	<code>tosca.datatypes.nfv.VirtualObjectStorageData</code>		Describes the object storage characteristics.

6.8.5.3 Attributes

None.

6.8.5.4 Requirements

None.

6.8.5.5 Capabilities

The capabilities of the `VirtualObjectStorage` node type shall comply with the provisions set out in table 6.8.5.5-1.

Table 6.8.5.5-1: Capabilities

Name	Type	Constraints	Description
virtual_storage	tosca.capabilities.nfv.VirtualStorage		Defines the capabilities of virtual_storage.

6.8.5.6 Definition

The syntax of the VirtualObjectStorage node type shall comply with the following definition:

```
tosca.nodes.nfv.Vdu.VirtualObjectStorage:
  derived_from: toska.nodes.Root
  description: This node type describes the specifications of requirements related
  to virtual object storage resources
  properties:
    virtual_object_storage_data:
      type: toska.datatypes.nfv.VirtualObjectStorageData
      description: Describes the object storage characteristics.
      required: true
  capabilities:
    virtual_storage:
      type: toska.capabilities.nfv.VirtualStorage
      description: Defines the capabilities of virtual_storage.
```

6.8.5.7 Additional requirements

None.

6.8.6 toska.nodes.nfv.Vdu.VirtualFileStorage

6.8.6.1 Description

The VirtualFileStorage node type describes the specifications of requirements related to virtual file storage resources, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.6.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.6.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualFileStorage
Type Qualified Name	toscanfv:VirtualFileStorage
Type URI	tosca.nodes.nfv.Vdu.VirtualFileStorage

6.8.6.2 Properties

The properties of the VirtualFileStorage node type shall comply with the provisions set out in table 6.8.6.2-1.

Table 6.8.6.2-1: Properties

Name	Required	Type	Constraints	Description
virtual_file_storage_data	yes	tosca.datatypes.nfv.VirtualFileStorageData		Describes the file storage characteristics.

6.8.6.3 Attributes

None.

6.8.6.4 Requirements

The requirements of the VirtualFileStorage node type shall comply with the provisions set out in table 6.8.6.4-1.

Table 6.8.6.4-1: Requirements

Name	Required	Type	Constraints	Description
virtual_link	yes	tosca.capabilities.nfv.VirtualLinkable		Describes the requirements for linking to virtual link

6.8.6.5 Capabilities

The capabilities of the VirtualFileStorage node type shall comply with the provisions set out in table 6.8.6.5-1.

Table 6.8.6.5-1: Capabilities

Name	Type	Constraints	Description
virtual_storage	tosca.capabilities.nfv.VirtualStorage		Defines the capabilities of virtual_storage.

6.8.6.6 Definition

The syntax of the VirtualFileStorage node type shall comply with the following definition:

```
tosca.nodes.nfv.Vdu.VirtualFileStorage:
  derived_from: tosca.nodes.Root
  description: This node type describes the specifications of requirements
  related to virtual file storage resources
  properties:
    virtual_file_storage_data:
      type: tosca.datatypes.nfv.VirtualFileStorageData
      description: Describes the file storage characteristics.
      required: true
  capabilities:
    virtual_storage:
      type: tosca.capabilities.nfv.VirtualStorage
      #description: Defines the capabilities of virtual_storage.
  requirements:
    - virtual_link:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      # description: Describes the requirements for linking to virtual link
```

6.8.6.7 Additional requirements

None.

6.8.7 tosca.nodes.nfv.Cp

6.8.7.1 Description

A Cp node type represents the Cpd information element as defined in ETSI GS NFV-IFA 011 [1], which describes network connectivity to a compute resource or a VL. This is an abstract type used as parent for the various Cp node types. Table 6.8.7.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.7.1-1: Type name, shorthand, and URI

Shorthand Name	Cp
Type Qualified Name	toscanfv:Cp
Type URI	tosca.nodes.nfv.Cp

6.8.7.2 Properties

The properties of the Cp node type shall comply with the provisions set out in table 6.8.7.2-1.

Table 6.8.7.2-1: Properties

Name	Required	Type	Constraints	Description
layer_protocols	yes	list of string	Valid values: ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire	Identifies which protocol the connection point uses for connectivity purposes.
role	no	string	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).
protocol	no	list of toasca.datatypes.nfv.CpProtocolData		Provides information on the addresses to be assigned to the connection point(s) instantiated from this Connection Point Descriptor.
trunk_mode	no	boolean		Information about whether the CP instantiated from this Cp is in Trunk mode (802.1Q or other). When operating in "trunk mode", the Cp is capable of carrying traffic for several VLANs. Absence of this property implies that trunkMode is not configured for the Cp i.e. It is equivalent to boolean value "false".

6.8.7.3 Attributes

None.

6.8.7.4 Requirements

None.

6.8.7.5 Capabilities

None.

6.8.7.6 Definition

The syntax of the Cp node type shall comply with the following definition:

```

tosca.nodes.nfv.Cp:
  derived_from: toska.nodes.Root
  description: Provides information regarding the purpose of the connection point
  properties:
    layer_protocols:
      type: list
      description: Identifies which protocol the connection point uses for connectivity
    purposes:
      required: true
      entry_schema:
        type: string
        constraints:
          - valid_values: [ ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire ]
    role: #Name in ETSI GS NFV-IFA 011 v0.7.3: cpRole
      type: string
      description: Identifies the role of the port in the context of the traffic flow
    patterns in the VNF or parent NS
      required: false
      constraints:
        - valid_values: [ root, leaf ]
      description:
        type: string
        description: Provides human-readable information on the purpose of the connection
    point
      required: false
    protocol:
      type: list
      description: Provides information on the addresses to be assigned to the
    connection point(s) instantiated from this Connection Point Descriptor
      required: false
    entry_schema:
      type: toska.datatypes.nfv.CpProtocolData
    trunk_mode:
      type: boolean
      description: Provides information about whether the CP instantiated from this Cp
    is in Trunk mode (802.1Q or other). When operating in "trunk mode", the Cp is capable
    of carrying traffic for several VLANs. Absence of this property implies that trunkMode
    is not configured for the Cp i.e. It is equivalent to boolean value "false".
      required: false

```

6.8.7.7 Additional requirements

The 'protocol' property shall not be included in a derived PnfExtCp node and shall be included in all other cases.

6.8.8 toska.nodes.nfv.VduCp

6.8.8.1 Description

A VduCp node type represents the VduCpd information element as defined in ETSI GS NFV-IFA 011 [1], which describes network connectivity between a VNFC instance (based on VDU) and an internal VL. Table 6.8.8.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.8.1-1: Type name, shorthand, and URI

Shorthand Name	VduCp
Type Qualified Name	toscanfv:VduCp
Type URI	tosca.nodes.nfv.VduCp

6.8.8.2 Properties

The properties of the VduCp node type shall comply with the provisions set out in table 6.8.8.2-1.

Table 6.8.8.2-1: Properties

Name	Required	Type	Constraints	Description
bitrate_requirement	no	integer	greater_or_equal: 0	Bitrate requirement in bit per second on this connection point.
virtual_network_interface_requirements	no	list of tosca.data types.nfv. VirtualNet workInterf aceRequir ements		Specifies requirements on a virtual network interface realising the CPs instantiated from this CPD.
order	no	integer	greater_or_equal: 0	The order of the NIC on the compute instance (e.g. eth2). See note. compute (aka multi vNICs) and ordering is desired, it is <i>*mandatory*</i> that all ports will be set with an order value and. The order values shall represent a positive, arithmetic progression that starts with 0 (e.g. 0, 1, 2, ..., n). If the property is not present, it shall be left to the VIM to assign the value when creating the instance.
vnic_type	no	string	Allowed values: normal, virtio, direct-physical	Describes the type of the virtual network interface realizing the CPs instantiated from this CPD. This is used to determine which mechanism driver(s) to be used to bind the port.

NOTE: When binding more than one port to a single.

6.8.8.3 Attributes

None.

6.8.8.4 Requirements

The requirements of the VduCp node type shall comply with the provisions set out in table 6.8.8.4-1.

Table 6.8.8.4-1: 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

6.8.8.5 Definition

The syntax of the VduCp node type shall comply with the following definition:

```

tosca.nodes.nfv.VduCp:
  derived_from: toska.nodes.nfv.Cp
  description: describes network connectivity between a VNFC instance based on this
  VDU and an internal VL
  properties:
    bitrate_requirement:
      type: integer # in bits per second
      description: Bitrate requirement in bit per second on this connection point
      required: false
      constraints:
        - greater_or_equal: 0
    virtual_network_interface_requirements:
      type: list
      description: Specifies requirements on a virtual network interface realising
      the CPs instantiated from this CPD
      required: false
      entry_schema:
        type: toska.datatypes.nfv.VirtualNetworkInterfaceRequirements
    order:
      type: integer
      description: The order of the NIC on the compute instance (e.g.eth2)
      required: false
      constraints:
        - greater_or_equal: 0
    vnic_type:
      type: string
      description: Describes the type of the virtual network interface realizing
      the CPs instantiated from this CPD
      required: false
      constraints:
        - valid_values: [ normal, virtio, direct-physical ]
    requirements:
      - virtual_link:
          capability: toska.capabilities.nfv.VirtualLinkable
          relationship: toska.relationships.nfv.VirtualLinksTo
      - virtual_binding:
          capability: toska.capabilities.nfv.VirtualBindable
          relationship: toska.relationships.nfv.VirtualBindsTo
          node: toska.nodes.nfv.Vdu.Compute

```

6.8.9 toska.nodes.nfv.VnfVirtualLink

6.8.9.1 Description

The VnfVirtualLink node type represents the VnfVirtualLinkDesc information element as defined in ETSI GS NFV-IFA 011 [1], which describes the information about an internal VNF VL. Table 6.8.9.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.9.1-1: Type name, shorthand, and URI

Shorthand Name	VnfVirtualLink
Type Qualified Name	toscanfv:VnfVirtualLink
Type URI	tosca.nodes.nfv.VnfVirtualLink

6.8.9.2 Properties

The properties of the VnfVirtualLink node type shall comply with the provisions set out in table 6.8.9.2-1.

Table 6.8.9.2-1: 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	list of string	Valid values: passive_monitoring, active_loopback	Test access facilities available on the VL.
vl_profile	yes	tosca.datatypes.nfv.VIProfile		Defines additional data for the VL: maximum and minimum bit rate requirements and QoS.
monitoring_parameters	no	list of tosca.datatypes.nfv.VirtualLinkMonitoringParameter		Describe monitoring parameters applicable to a VL instantiated from this node type.

6.8.9.3 Requirements

None.

6.8.9.4 Capabilities

The capabilities of the VnfVirtualLink node type shall comply with the provisions set out in table 6.8.9.4-1.

Table 6.8.9.4-1: Capabilities

Name	Type	Constraints	Description
virtual_linkable	tosca.capabilities.nfv.VirtualLinkable		Defines ability of VirtualLinkable.

6.8.9.5 Definition

The syntax of the VnfVirtualLink node type shall comply with the following definition:

```

tosca.nodes.nfv.VnfVirtualLink:
  derived_from: toska.nodes.Root
  description: Describes the information about an internal VNF VL
  properties:
    connectivity_type:
      type: toska.datatypes.nfv.ConnectivityType
      description: Specifies the protocol exposed by the VL and the flow pattern
supported by the VL
      required: true
    description:
      type: string
      description: Provides human-readable information on the purpose of the VL
      required: false
    test_access:
      type: list
      description: Test access facilities available on the VL
      required: false

```

```

entry_schema:
  type: string
  constraints:
    - valid_values: [ passive_monitoring, active_loopback ]
vl_profile:
  type: toska.datatypes.nfv.VlProfile
  description: Defines additional data for the VL
  required: true
monitoring_parameters:
  type: list
  entry_schema:
    type: toska.datatypes.nfv.VirtualLinkMonitoringParameter
    description: Describes monitoring parameters applicable to the VL
    required: false
capabilities:
  virtual_linkable:
    type: toska.capabilities.nfv.VirtualLinkable

```

6.9 Group Types

6.9.1 toska.groups.nfv.PlacementGroup

6.9.1.1 Description

PlacementGroup is used for describing the affinity or anti-affinity relationship applicable between the virtualization containers to be created based on different VDUs, or between internal VLs to be created based on different VnfVirtualLinkDesc(s). Table 6.9.1.1-1 specifies the declared names for this group type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.9.1.1-1: Type name, shorthand, and URI

Shorthand Name	PlacementGroup
Type Qualified Name	toscanfv:PlacementGroup
Type URI	tosca.groups.nfv.PlacementGroup

6.9.1.2 Properties

The properties of the PlacementGroup group type shall comply with the provisions set out in table 6.9.1.2-1.

Table 6.9.1.2-1: Properties

Name	Required	Type	Constraints	Description
description	yes	string		Human readable description of the group

6.9.1.3 Definition

The syntax of the PlacementGroup group type shall comply with the following definition:

```

tosca.groups.nfv.PlacementGroup:
  derived_from: toska.groups.Root
  description: PlacementGroup is used for describing the affinity or anti-affinity
relationship applicable between the virtualization containers to be created based on
different VDUs, or between internal VLs to be created based on different
VnfVirtualLinkDesc(s)
  properties:
    description:

```

```

type: string
description: Human readable description of the group
required: true
members: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink ]

```

6.9.1.4 Additional Requirements

A group with type `tosca.groups.nfv.PlacementGroup` shall contain more than one member when used as the target of an `AffinityRule` or `AntiAffinityRule` policy.

6.9.1.5 Examples

See clause 6.10.10.5.

6.10 Policy Types

6.10.1 `tosca.policies.nfv.InstantiationLevels`

6.10.1.1 Description

The `InstantiationLevels` type is a policy type representing all the instantiation levels of resources to be instantiated within a deployment flavour and including default instantiation level in term of the number of VNFC instances to be created as defined in ETSI GS NFV-IFA 011 [1]. This policy concerns the whole VNF (deployment flavour) represented by the `topology_template` and thus has no explicit target list. Table 6.10.1.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.1.1-1: Type name, shorthand, and URI

Shorthand Name	<code>InstantiationLevels</code>
Type Qualified Name	<code>toscanfv:InstantiationLevels</code>
Type URI	<code>tosca.policies.nfv.InstantiationLevels</code>

6.10.1.2 Properties

The properties of the `InstantiationLevels` policy type shall comply with the provisions set out in table 6.10.1.2-1.

Table 6.10.1.2-1: Properties

Name	Required	Type	Constraints	Description
<code>levels</code>	yes	map of <code>tosca.datatypes.nfv.InstantiationLevel</code>		Describes the various levels of resources that can be used to instantiate the VNF using this flavour.
<code>default_level</code>	no	string		The default instantiation level for this flavour.

6.10.1.3 Definition

The syntax of the `InstantiationLevels` policy type shall comply with the following definition:

```

tosca.policies.nfv.InstantiationLevels:
  derived_from: tosca.policies.Root
  description: The InstantiationLevels type is a policy type representing all the
instantiation levels of resources to be instantiated within a deployment flavour and

```

including default instantiation level in term of the number of VNFC instances to be created as defined in ETSI GS NFV-IFA 011 [1].

```

properties:
  levels:
    type: map # key: levelId
    description: Describes the various levels of resources that can be used to
instantiate the VNF using this flavour.
    required: true
    entry_schema:
      type: tosca.datatypes.nfv.InstantiationLevel
  default_level:
    type: string # levelId
    description: The default instantiation level for this flavour.
    required: false # required if multiple entries in levels

```

6.10.2 tosca.policies.nfv.VduInstantiationLevels

6.10.2.1 Description

The VduInstantiationLevels type is a policy type representing all the instantiation levels of resources to be instantiated within a deployment flavour in term of the number of VNFC instances to be created from each vdu.Compute. as defined in ETSI GS NFV-IFA 011 [1].

Table 6.10.2.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.2.1-1: Type name, shorthand, and URI

Shorthand Name	VduInstantiationLevels
Type Qualified Name	toscanfv:VduInstantiationLevels
Type URI	tosca.policies.nfv.VduInstantiationLevels

6.10.2.2 Properties

The properties of the VduInstantiationLevels policy type shall comply with the provisions set out in table 6.10.2.2-1.

Table 6.10.2.2-1: Properties

Name	Required	Type	Constraints	Description
levels	yes	map of tosca.d atatype s.nfv.Vd uLevel		Describes the Vdu.Compute levels of resources that can be used to instantiate the VNF using this flavour.

6.10.2.3 Definition

The syntax of the VduInstantiationLevels policy type shall comply with the following definition:

```

tosca.policies.nfv.VduInstantiationLevels:
  derived_from: tosca.policies.Root
  description: The VduInstantiationLevels type is a policy type representing all the
instantiation levels of resources to be instantiated within a deployment flavour in
term of the number of VNFC instances to be created from each vdu.Compute. as defined
in ETSI GS NFV-IFA 011 [1]
  properties:
    levels:

```

```

type: map # key: levelId
description: Describes the Vdu.Compute levels of resources that can be used to
instantiate the VNF using this flavour
required: true
entry_schema:
  type: tosca.datatypes.nfv.VduLevel
targets: [ tosca.nodes.nfv.Vdu.Compute ]

```

6.10.2.4 Additional Requirements

A VduInstantiationLevels policy shall contain an entry for each instantiation level (and only for them) defined in the InstantiationLevels policy.

6.10.3 tosca.policies.nfv.VirtualLinkInstantiationLevels

6.10.3.1 Description

The VirtualLinkInstantiationLevels type is a policy type representing all the instantiation levels of virtual link resources to be instantiated within a deployment flavour as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.3.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.3.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualLinkInstantiationLevels
Type Qualified Name	toscanfv:VirtualLinkInstantiationLevels
Type URI	tosca.policies.nfv.VirtualLinkInstantiationLevels

6.10.3.2 Properties

The properties of the VirtualLinkInstantiationLevels policy type shall comply with the provisions set out in table 6.10.3.2-1.

Table 6.10.3.2-1: Properties

Name	Required	Type	Constraints	Description
levels	yes	map of tosca.datatypes.nfv.VirtualLinkBitrateLevel		Describes the virtual link levels of resources that can be used to instantiate the VNF using this flavour.

6.10.3.3 Definition

The syntax of the VirtualLinkInstantiationLevels policy type shall comply with the following definition:

```

tosca.policies.nfv.VirtualLinkInstantiationLevels:
  derived_from: tosca.policies.Root
  description: The VirtualLinkInstantiationLevels type is a policy type representing
all the instantiation levels of virtual link resources to be instantiated within a
deployment flavour as defined in ETSI GS NFV-IFA 011 [1].
  properties:
    levels:
      type: map # key: levelId
      description: Describes the virtual link levels of resources that can be used to
instantiate the VNF using this flavour.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.VirtualLinkBitrateLevel

```

```
targets: [ tosca.nodes.nfv.VnfVirtualLink ]
```

6.10.3.4 Additional Requirements

A VirtualLinkInstantiationLevels policy shall contain an entry for each instantiation level (and only for them) defined in the InstantiationLevels policy.

6.10.4 Void

6.10.5 tosca.policies.nfv.ScalingAspects

6.10.5.1 Description

The ScalingAspects type is a policy type representing the scaling aspects used for horizontal scaling as defined in ETSI GS NFV-IFA 011 [1]. This policy concerns the whole VNF (deployment flavour) represented by the topology_template and thus has no explicit target list. Table 6.10.5.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.5.1-1: Type name, shorthand, and URI

Shorthand Name	ScalingAspects
Type Qualified Name	toscanfv:ScalingAspects
Type URI	tosca.policies.nfv.ScalingAspects

6.10.5.2 Properties

The properties of the ScalingAspects policy type shall comply with the provisions set out in table 6.10.5.2-1.

Table 6.10.5.2-1: Properties

Name	Required	Type	Constraints	Description
aspects	yes	Map of tosca.datatypes.nfv. ScalingAspect		Describe maximum scale level for total number of scaling steps that can be applied to a particular aspect.

6.10.5.3 Definition

The syntax of the ScalingAspects policy type shall comply with the following definition:

```
tosca.policies.nfv.ScalingAspects:
  derived_from: tosca.policies.Root
  description: The ScalingAspects type is a policy type representing the scaling
  aspects used for horizontal scaling as defined in ETSI GS NFV-IFA 011 [1]
  properties:
    aspects:
      type: map # key: aspectId
      description: Describe maximum scale level for total number of scaling steps that
      can be applied to a particular aspect
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.ScalingAspect
```

6.10.5.4 Examples

See clause A.6.

6.10.6 `tosca.policies.nfv.VduScalingAspectDeltas`

6.10.6.1 Description

The `VduScalingAspectDeltas` type is a policy type representing the `Vdu.Compute` detail of an aspect deltas used for horizontal scaling, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.6.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.6.1-1: Type name, shorthand, and URI

Shorthand Name	<code>VduScalingAspectDeltas</code>
Type Qualified Name	<code>toscanfv.VduScalingAspectDeltas</code>
Type URI	<code>tosca.policies.nfv.VduScalingAspectDeltas</code>

6.10.6.2 Properties

The properties of the `VduScalingAspectDeltas` policy type shall comply with the provisions set out in table 6.10.6.2-1.

Table 6.10.6.2-1: Properties

Name	Required	Type	Constraints	Description
<code>aspect:</code>	yes	string		Represents the scaling aspect to which this policy applies.
<code>deltas</code>	yes	map of <code>tosca.datatypes.nfv.VduLevel</code>		Describes the <code>Vdu.Compute</code> scaling deltas to be applied for every scaling steps of a particular aspect.

6.10.6.3 Definition

The syntax of the `VduScalingAspectDeltas` policy type shall comply with the following definition:

```
tosca.policies.nfv.VduScalingAspectDeltas:
  derived_from: toasca.policies.Root
  description: The VduScalingAspectDeltas type is a policy type representing the
  Vdu.Compute detail of an aspect deltas used for horizontal scaling, as defined in ETSI
  GS NFV-IFA 011 [1]
  properties:
    aspect:
      type: string
      description: Represents the scaling aspect to which this policy applies
      required: true
    deltas:
      type: map # key: scalingDeltaId
      description: Describes the Vdu.Compute scaling deltas to be applied for every
      scaling steps of a particular aspect.
      required: true
      entry_schema:
        type: toasca.datatypes.nfv.VduLevel
      targets: [ toasca.nodes.nfv.Vdu.Compute ]
```

6.10.6.4 Examples

See clause A.6.

6.10.7 tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas

6.10.7.1 Description

The VirtualLinkBitrateScalingAspectDeltas type is a policy type representing the VnfVirtualLink detail of an aspect deltas used for horizontal scaling, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.7.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.7.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualLinkBitrateScalingAspectDeltas
Type Qualified Name	toscanfv:VirtualLinkBitrateScalingAspectDeltas
Type URI	tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas

6.10.7.2 Properties

The properties of the VirtualLinkBitrateScalingAspectDeltas policy type shall comply with the provisions set out in table 6.10.7.2-1.

Table 6.10.7.2-1: Properties

Name	Required	Type	Constraints	Description
aspect	yes	string		Represents the scaling aspect to which this policy applies.
aspect_deltas	yes	map of tosca.datatypes.nfv.VirtualLinkBitrateLevel		Describes the VnfVirtualLink scaling deltas to be applied for every scaling steps of a particular aspect.

6.10.7.3 Definition

The syntax of the VirtualLinkBitrateScalingAspectDeltas policy type shall comply with the following definition:

```
tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas:
  derived_from: tosca.policies.Root
  description: The VirtualLinkBitrateScalingAspectDeltas type is a policy type
  representing the VnfVirtualLink detail of an aspect deltas used for horizontal
  scaling, as defined in ETSI GS NFV-IFA 011 [1].
  properties:
    aspect:
      type: string
      description: Represents the scaling aspect to which this policy applies.
      required: true
    deltas:
      type: map # key: scalingDeltaId
      description: Describes the VnfVirtualLink scaling deltas to be applied for every
      scaling steps of a particular aspect.
      required: true
  entry_schema:
    type: tosca.datatypes.nfv.VirtualLinkBitrateLevel
  targets: [ tosca.nodes.nfv.VnfVirtualLink ]
```

6.10.7.4 Examples

See clause A.6.

6.10.8 tosca.policies.nfv.VduInitialDelta

6.10.8.1 Description

The VduInitialDelta type is a policy type representing the Vdu.Compute detail of an initial delta used for horizontal scaling, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.8.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.8.1-1: Type name, shorthand, and URI

Shorthand Name	VduInitialDelta
Type Qualified Name	toscanfv:VduInitialDelta
Type URI	tosca.policies.nfv.VduInitialDelta

6.10.8.2 Properties

The properties of the VduInitialDelta policy type shall comply with the provisions set out in table 6.10.8.2-1.

Table 6.10.8.2-1: Properties

Name	Required	Type	Constraints	Description
initial_delta:	yes	tosca.datatypes.nfv.VduLevel		Represents the initial minimum size of the VNF.

6.10.8.3 Definition

The syntax of the VduInitialDelta policy type shall comply with the following definition:

```
tosca.policies.nfv.VduInitialDelta:
  derived_from: tosca.policies.Root
  description: The VduInitialDelta type is a policy type representing the
  Vdu.Compute detail of an initial delta used for horizontal scaling, as defined in
  ETSI GS NFV-IFA 011 [1].
  properties:
    initial_delta:
      type: tosca.datatypes.nfv.VduLevel
      description: Represents the initial minimum size of the VNF.
      required: true
  targets: [ tosca.nodes.nfv.Vdu.Compute ]
```

6.10.8.4 Examples

See clause A.6.

6.10.9 tosca.policies.nfv.VirtualLinkBitrateInitialDelta

6.10.9.1 Description

The VirtualLinkBitrateInitialDelta type is a policy type representing the VnfVirtualLink detail of an initial deltas used for horizontal scaling, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.9.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.9.1-1: Type name, shorthand, and URI

Shorthand Name	VirtualLinkBitrateInitialDelta
Type Qualified Name	toscanfv:VirtualLinkBitrateInitialDelta
Type URI	tosca.policies.nfv.VirtualLinkBitrateInitialDelta

6.10.9.2 Properties

The properties of the VirtualLinkBitrateInitialDelta policy type shall comply with the provisions set out in table 6.10.9.2-1.

Table 6.10.9.2-1: Properties

Name	Required	Type	Constraints	Description
initial_delta:	yes	tosca.datatypes.nfv.VirtualLinkBitrateLevel		Represents the initial minimum size of the VNF.

6.10.9.3 Definition

The syntax of the VirtualLinkBitrateInitialDelta policy type shall comply with the following definition:

```
tosca.policies.nfv.VirtualLinkBitrateInitialDelta:
  derived_from: toasca.policies.Root
  description: The VirtualLinkBitrateInitialDelta type is a policy type
  representing the VnfVirtualLink detail of an initial deltas used for horizontal
  scaling, as defined in ETSI GS NFV-IFA 011 [1].
  properties:
    initial_delta:
      type: toasca.datatypes.nfv.VirtualLinkBitrateLevel
      description: Represents the initial minimum size of the VNF.
      required: true
  targets: [ toasca.nodes.nfv.VnfVirtualLink ]
```

6.10.9.4 Examples

See clause A.6.

6.10.10 AffinityRule, AntiAffinityRule

6.10.10.1 Description

The AffinityRule or AntiAffinityRule describes the affinity or anti-affinity rules applicable for the defined targets:

- If there is only one node template with node type toasca.nodes.nfv.Vdu.Compute or toasca.nodes.nfv.VnfVirtualLink set as the targets, the AffinityRule or AntiAffinityRule applies between the virtualization containers to be created based on a particular VDU, or between internal VLs to be created based on a particular VnfVirtualLinkDesc as described in ETSI GS NFV-IFA 011 [1].
- If there are more than one node templates with node type toasca.nodes.nfv.Vdu.Compute or toasca.nodes.nfv.VnfVirtualLink set as the targets, or a group with type toasca.groups.nfv.PlacementGroup which contains more than one members set as targets, the AffinityRule or AntiAffinityRule applies between the virtualization containers to be created based on different VDUs, or between internal VLs to be created based on different VnfVirtualLinkDesc(s) as described in ETSI GS NFV-IFA 011 [1].

Table 6.10.10.1-1 and 6.10.10.1-2 specify the declared names for the policy types. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.10.1-1: Type name, shorthand, and URI

Shorthand Name	AffinityRule
Type Qualified Name	toscanfv:AffinityRule
Type URI	tosca.policies.nfv.AffinityRule

Table 6.10.10.1-2: Type name, shorthand, and URI

Shorthand Name	AntiAffinityRule
Type Qualified Name	toscanfv:AntiAffinityRule
Type URI	tosca.policies.nfv.AntiAffinityRule

6.10.10.2 Properties

The properties of the AffinityRule and AntiAffinityRule types shall comply with the provisions set out in table 6.10.10.2-1.

Table 6.10.10.2-1: Properties

Name	Required	Type	Constraints	Description
scope	Yes	String	Possible values are "nfvi_pop", "zone", "zone_group", "nfvi_node".	Specifies the scope of the local affinity rule

6.10.10.3 targets

The targets of the AffinityRule and AntiAffinityRule policy types shall comply with the provisions set out in table 6.10.10.3-1.

Table 6.10.10.3-1: Targets

Name	Required	Type	Constraints	Description
targets	Yes	string[]	Valid types: tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink, tosca.groups.nfv.PlacementGroup	In case of LocalAffinityOrAntiAffinityRule as defined in ETSI GS NFV-IFA 011 [1], the valid type of the targets is tosca.nodes.nfv.Vdu.Compute or tosca.nodes.nfv.VnfVirtualLink. In case of affinityOrAntiAffinityGroup as defined in ETSI GS NFV-IFA 011 [1], the valid types of the targets are: tosca.nodes.nfv.Vdu.Compute and tosca.nodes.nfv.VnfVirtualLink or a tosca.groups.nfv.PlacementGroup.

6.10.10.4 Definition

The syntax of the AffinityRule policy type shall comply with the following definition:

```
tosca.policies.nfv.AffinityRule:
  derived_from: tosca.policies.Placement
  description: The AffinityRule describes the affinity rules applicable for the
  defined targets
  properties:
    scope:
      type: string
      description: scope of the rule is an NFVI_node, an NFVI_PoP, etc.
      required: true
```

```

constraints:
  - valid_values: [ nfvi_node, zone, zone_group, nfvi_pop ]
targets: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink,
tosca.groups.nfv.PlacementGroup ]

```

The syntax of the AntiAffinityRule policy type shall comply with the following definition:

```

tosca.policies.nfv.AntiAffinityRule:
  derived_from: tosca.policies.Placement
  description: The AntiAffinityRule describes the anti-affinity rules applicable for
the defined targets
  properties:
    scope:
      type: string
      description: scope of the rule is an NFVI_node, an NFVI_PoP, etc.
      required: true
    constraints:
      - valid_values: [ nfvi_node, zone, zone_group, nfvi_pop ]
  targets: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink,
tosca.groups.nfv.PlacementGroup ]

```

6.10.10.5 Examples

The following example template fragments illustrate the concepts:

```

node_templates:
  VDU_1:
    type: tosca.nodes.nfv.Vdu.Compute

policies:
  policy_affinity_local_VDU_1:
    type: tosca.policies.nfv.AffinityRule
    targets: [ VDU_1 ]
    properties:
      scope: nfvi_node

```

The above example illustrates a local affinity rule for VDU_1.

```

node_template:
  VDU_1:
    type: tosca.nodes.nfv.Vdu.Compute

  VDU_2:
    type: tosca.nodes.nfv.Vdu.Compute

groups:
  affinityOrAntiAffinityGroup_1:
    type: tosca.groups.nfv.PlacementGroup
    members: [ VDU_1, VDU_2 ]

policies:
  policy_antiaffinity_group_1:
    type: tosca.policies.nfv.AntiAffinityRule
    targets: [ affinityOrAntiAffinityGroup_1 ]
    properties:
      scope: nfvi_node

```

The above example illustrates an anti-affinity policy among a group which contains VDU_1 and VDU_2 as members.

6.10.11 toasca.policies.nfv.SecurityGroupRule

6.10.11.1 Description

The SecurityGroupRule type is a policy type specifying the matching criteria for the ingress and/or egress traffic to and from visited connection points as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.11.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.11.1-1: Type name, shorthand, and URI

Shorthand Name	SecurityGroupRule
Type Qualified Name	toscanfv:SecurityGroupRule
Type URI	tosca.policies.nfv.SecurityGroupRule

6.10.11.2 Properties

The properties of the SecurityGroupRule policy type shall comply with the provisions set out in table 6.10.11.2-1.

Table 6.10.11.2-1: Properties

Name	Required	Type	Constraints	Description
description	no	string		Human readable description of the security group rule.
direction	no	string	ingress, egress	The direction in which the security group rule is applied. The direction of 'ingress' or 'egress' is specified against the associated CP. I.e., 'ingress' means the packets entering a CP, while 'egress' means the packets sent out of a CP.
ether_type	no	string	ipv4, ipv6	Indicates the protocol carried over the Ethernet layer.
protocol	no	string	see note	Indicates the protocol carried over the IP layer. Permitted values: any protocol defined in the IANA protocol registry [i.5], e.g. TCP, UDP, ICMP, etc.
port_range_min	no	integer	0 - 65535	Indicates minimum port number in the range that is matched by the security group rule. If a value is provided at design-time, this value may be overridden at run-time based on other deployment requirements or constraints.
port_range_max	no	integer	0 – 65535	Indicates maximum port number in the range that is matched by the security group rule. If a value is provided at design-time, this value may be overridden at run-time based on other deployment requirements or constraints.
NOTE:	"protocol" constraints values: hopopt, icmp, igmp, ggp, ipv4, st, tcp, cbt, egg, igp, bbn_rcc_mon, nvp_ii, pup, argus, emcon, xnet, chaos, udp, mux, dcn_meas, hmp, prm, xns_idp, trunk_1, trunk_2, leaf_1, leaf_2, rdp, irtp, iso_tp4, netblt, mfe_nsp, merit_inp, dccp, 3pc, idpr, xtp, ddp, idpr_cmt, tp++, il, ipv6, sdrp, ipv6_route, ipv6_frag, idrp, rsvp, gre, dsr, bna, esp, ah, i_nlsp, swipe, narp, mobile, tlsp, skip, ipv6_icmp, ipv6_no_nxt, ipv6_opts, cftp, sat_expak, kryptolan, rvd, ippc, sat_mon, visa, ipcv, cpnx, cphb, wsn, pvp, br_sat_mon, sun_nd, wb_mon, wb_expak, iso_ip, vmtp, secure_vmtp, vines, ttp, iptm, nsfnet_igp, dgp, tcf, eigrp, ospfigp, sprite_rpc, larp, mtp, ax.25, ipip, micp, scc_sp, etherip, encap, gmtp, ifmp, pnri, pim, aris, scps, qnx, a/n, ip_comp, snp, compaq_peer, ipx_in_ip, vrrp, pgm, l2tp, ddx, iatp, stp, srp, uti, smp, sm, ptp, isis, fire, crtp, crudp, sscopmce, iplt, sps, pipe, sctp, fc, rsvp_e2e_ignore, mobility, udp_lite, mpls_in_ip, manet, hip, shim6, wesp, rohc.			

6.10.11.3 Definition

The syntax of the SecurityGroupRule policy type shall comply with the following definition:

```

tosca.policies.nfv.SecurityGroupRule:
  derived_from: tosca.policies.Root
  description: The SecurityGroupRule type is a policy type specified the matching
  criteria for the ingress and/or egress traffic to/from visited connection points
  as defined in ETSI GS NFV-IFA 011 [1].
  properties:
    description:
      type: string
      description: Human readable description of the security group rule.
      required: false
    direction:
      type: string
      description: The direction in which the security group rule is applied. The
      direction of 'ingress' or 'egress' is specified against the associated CP. I.e.,
      'ingress' means the packets entering a CP, while 'egress' means the packets sent
      out of a CP.
      required: false
      constraints:
        - valid_values: [ ingress, egress ]
      default: ingress
    ether_type:
      type: string
      description: Indicates the protocol carried over the Ethernet layer.
      required: false
      constraints:
        - valid_values: [ ipv4, ipv6 ]
      default: ipv4
    protocol:
      type: string
      description: Indicates the protocol carried over the IP layer. Permitted
      values include any protocol defined in the IANA protocol registry, e.g. TCP, UDP,
      ICMP, etc.
      required: false
      constraints:
        - valid_values: [ hopopt, icmp, igmp, ggp, ipv4, st, tcp, cbt, egp, igp,
        bbn_rcc_mon, nvp_ii, pup, argus, emcon, xnet, chaos, udp, mux, dcn_meas, hmp, prm,
        xns_idp, trunk_1, trunk_2, leaf_1, leaf_2, rdp, irtp, iso_tp4, netblt, mfe_nsp,
        merit_inp, dccp, 3pc, idpr, xtp, ddp, idpr_cmt, tp++, il, ipv6, sdrp, ipv6_route,
        ipv6_frag, idrp, rsvp, gre, dsr, bna, esp, ah, i_nls, swipe, nar, mobile, tisp,
        skip, ipv6_icmp, ipv6_no_nxt, ipv6_opts, cftp, sat_expak, kryptolan, rvd, ippc,
        sat_mon, visa, ipc, cpnx, cphb, wsn, pvp, br_sat_mon, sun_nd, wb_mon, wb_expak,
        iso_ip, vmtp, secure_vmtp, vines, ttp, iptm, nsfnet_igp, dgp, tcf, eigrp, ospfigp,
        sprite_rpc, larp, mtp, ax.25, ipip, micp, scc_sp, etherip, encap, gmtp, ifmp,
        pnni, pim, aris, scps, qnx, a/n, ip_comp, snp, compaq_peer, ipx_in_ip, vrrp, pgm,
        l2tp, ddx, iatp, stp, srp, uti, smp, sm, ptp, isis, fire, crtp, crudp, sscopmce,
        iplt, sps, pipe, sctp, fc, rsvp_e2e_ignore, mobility, udp_lite, mpls_in_ip, manet,
        hip, shim6, wesp, rohc ]
      default: tcp
    port_range_min:
      type: integer
      description: Indicates minimum port number in the range that is matched by the
      security group rule. If a value is provided at design-time, this value may be
      overridden at run-time based on other deployment requirements or constraints.
      required: false
      constraints:
        - greater_or_equal: 0

```

```

- less_or_equal: 65535
default: 0
port_range_max:
type: integer
description: Indicates maximum port number in the range that is matched by the
security group rule. If a value is provided at design-time, this value may be
overridden at run-time based on other deployment requirements or constraints.
required: false
constraints:
- greater_or_equal: 0
- less_or_equal: 65535
default: 65535
targets: [ tosca.nodes.nfv.VduCp, tosca.nodes.nfv.VnfExtCp ]

```

6.10.11.4 Additional Requirements

If an ingress traffic is accepted, the corresponding egress response shall be allowed, regardless of the security group rules on the egress direction. If an egress traffic is allowed, the corresponding ingress response shall be accepted, regardless of the security group rules on the ingress direction. The design of security group rule follows a permissive model where all security group rules applied to a connection point are dealt with in an "OR" logic fashion, i.e. the traffic is allowed if it matches any security group rule applied to this connection point.

6.10.12 tosca.policies.nfv.SupportedVnfInterface

6.10.12.1 Description

The SupportedVnfInterface policy type represents interfaces produced by a VNF, the details to access them and the applicable connection points to use to access these interfaces. It corresponds to the VnfInterfaceDetails information element defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.12.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.12.1-1: Type name, shorthand, and URI

Shorthand Name	SupportedVnfInterface
Type Qualified Name	toscanfv:SupportedVnfInterface
Type URI	tosca.policies.nfv.SupportedVnfInterface

6.10.12.2 Properties

The properties of the SupportedVnfInterface policy type shall comply with the provisions set out in table 6.10.12.2-1.

Table 6.10.12.2-1: Properties

Name	Required	Type	Constraints	Description
interface_name	yes	string	Valid values vnf_indicator vnf_configuration	Identifies an interface produced by the VNF.
details	no	tosca.datatypes.nfv.InterfaceDetails		Provide additional data to access the interface endpoint (e.g. API URI prefix).

6.10.12.3 Definition

The syntax of the SupportedVnfInterface policy type shall comply with the following definition:

```

tosca.policies.nfv.SupportedVnfInterface:
  derived_from: tosca.policies.Root

```

```

description: this policy type represents interfaces produced by a VNF, the
details to access them and the applicable connection points to use to access these
interfaces
properties:
  interface_name:
    type: string
    description: Identifies an interface produced by the VNF.
    required: true
    constraints:
      - valid_values: [ vnf_indicator, vnf_configuration ]
  details:
    type: toska.datatypes.nfv.InterfaceDetails
    description: Provide additional data to access the interface endpoint
    required: false
targets: [ toska.nodes.nfv.VnfExtCp, toska.nodes.nfv.VduCp ]

```

6.10.12.4 Additional requirements

The valid targets for this policy type shall be the node templates representing the connection point descriptors from which to instantiate the connection point instances through which the interfaces can be accessed. This may be a VnfExtCp node template or a VduCp node template when an internal connection point is re-exposed externally.

6.10.12.5 Example

```

policies:
  policy_interface_1:
    type: toska.policies.nfv.SupportedVnfInterface
    targets: [ MyVnfmFacingExtCp ]
    properties:
      interface_name: vnf_indicator
    details:
      uri_components:
        scheme: https
        authority:
          host: myvnf.example.com

```

6.11 VNFD TOSCA service template design

6.11.1 General

The TOSCA service template design for a VNFD in the general case uses two levels of service templates as described in clause 6.11.2. In this design, the top level contains an abstract node template, i.e. without an implementation of the creation operation and is therefore substituted by one of the lower level service templates. This design is applicable regardless of whether the VNF has one or multiple deployment flavours.

In the particular case of a VNF with only one deployment flavour there is an alternative design which is described in clause 6.11.3 and which uses only one service template.

6.11.2 Single or multiple deployment flavour design with TOSCA-Simple-Profile-YAML-v1.2

VNFD shall be implemented as one top-level service template and one or multiple lower level service templates, where each lower level service template represents a deployment flavour. In addition, an abstract node type representing the VNF with multiple deployment flavours is also required. The top level service template shall be the main entry point of the VNF package, i.e. the Entry-definitions file.. The node type definition is included in a separate YAML file imported from the top and from the lower level files.

See clause A.2 for an example of VNFD design with multiple deployment flavours.

The top level service template shall comply with TOSCA-Simple-Profile-YAML-v1.2 [3] and shall include:

- a) an import statement referencing the definitions of all the TOSCA types specified in the present document;
- b) an import statement referencing the definitions of all low level service template(s);
- c) an import statement referencing a yaml file which contains a VNF specific node type definition which shall be derived from `tosca.nodes.nfv.VNF` node type as defined in clause 6.8.1; and
- d) a topology template with a node template of the VNF specific node type, which shall include the `flavour_id` and other properties and the requirements as defined in clause 6.8.1.

The deployment and lifecycle management of instances of this VNF node type is done by means of substitution by any of the lower level service templates.

The lower level service template is an implementable TOSCA service template for the deployment of a specific deployment flavour. The lower level service template shall comply with TOSCA-Simple-Profile-YAML-v1.2 [3] and shall include:

- a) an import statement referencing the definitions of all TOSCA types specified in clause 6 of the present document;
- b) an import statement referencing a yaml file which contains a VNF specific node type definition which shall be derived from `tosca.nodes.nfv.VNF` node type as defined in clause 6.8.1; and
- c) a topology template describing the internal topology of the VNF with:
 - `substitution_mappings` indicating:
 - the same node type as defined in the top level service template;
 - a `flavour_id` property value which identifies the DF corresponding to this low level template within the VNFD; and
 - the mapping of the `virtual_link` requirements on external connection points;
 - a node template referencing the VNF specific node type, implementations of the operations of the LCM interface to be executed by the VNFM, if applicable; and
 - additional node templates of type `Vdu.Compute`, `Vdu.VirtualBlockStorage`, `Vdu.VirtualObjectStorage`, `Vdu.VirtualFileStorage` `VduCp`, etc. that define the topology and composition of the VNF flavour.

The yaml file with the VNF specific node type definition shall be included in the VNF package.

NOTE: The format and structure of a VNF package is defined in ETSI GS NFV-SOL 004 [i.6].

When the `flavour_id` of a VNF has been chosen (e.g. through an input parameter of a VNF instantiation request received by a VNFM) among the values included in the VNF node type imported into the top level service template, it is then used as the property constraint for selecting a particular lower level TOSCA service template inside the VNFD by using abstract node template matching with substitution as described in TOSCA-Simple-Profile-YAML-v1.2 [3].

6.11.3 Single deployment flavour design with TOSCA-Simple-Profile-YAML-v1.1

In case of single deployment flavour scenario, TOSCA-Simple-Profile-YAML-v1.1 [4] may be used to describe the TOSCA service templates.

NOTE 1: The present document does not preclude this pattern using TOSCA-Simple-Profile-YAML-v1.2 [3]. In this case one single service template is used as specified in TOSCA-Simple-Profile-YAML-v1.1 [4] and includes:

- a) an import statement referencing the definitions of all TOSCA types specified in clause 6 of the present document;
- b) an abstract VNF node type definition derived from the `tosca.nodes.nfv.VNF` node type, as defined in clause 6.8.1; and
- c) a topology template describing the internal topology of the VNF with:
 - `substitution_mappings` indicating the same abstract VNF node type and the mapping of the `virtual_link` requirements on connection points;
 - a node template of this abstract VNF node type with the `flavour_id` and other properties and, if applicable, implementations of the operations of the LCM interface to be executed by the VNFM; and
 - additional node templates of type `Vdu.Compute`, `Vdu.VirtualBlockStorage`, `Vdu.VirtualObjectStorage`, `Vdu.VirtualFileStorage`, `VduCp`, etc. that define the topology and composition of the VNF flavour.

See clause A.5 for an example of VNFD design with single deployment flavour.

NOTE 2: The service template is deployed stand-alone, i.e. without performing a substitution. However including the `substitution_mappings` rule indicate its ability to substitute a node template of the abstract VNF node type, which may appear in an NSD.

7 NSD TOSCA model

7.1 Introduction

The NSD information model specified by ETSI GS NFV-IFA 014 [2] is mapped to the TOSCA concepts. NSD occurrences are represented as TOSCA service templates, as defined in the TOSCA Simple YAML Profile v1.2 [3], to be used by the NFVO for managing the lifecycle of NS instances.

Table 7.1-1 shows an overview of the mapping between the main NSD information elements defined in ETSI GS NFV-IFA 014 [2] and TOSCA types defined in the present document. The definition of all TOSCA types for representing all information elements is described in the following clauses.

NOTE: There is only part of NSD features as specified in ETSI GS NFV-IFA 014 [2] included in this version of the present document, the following are features not included:

- Multiple NS deployment flavours
- NS scaling Aspect
- Multiple Ns instantiation levels
- Nested NS design
- Vnffgd
- NS Monitoring

- VnfIndicators
- Dependencies

Table 7.1-1: Mapping of IFA014 information elements with TOSCA types

ETSI NFV Information Element ETSI GS NFV-IFA 014 [2]	TOSCA type	Derived from
NSD	tosca.nodes.nfv.NS	tosca.nodes.Root
Sapd	tosca.nodes.nfv.Sap	tosca.nodes.Root
NsVirtualLinkDesc	tosca.nodes.nfv.NsVirtualLink	tosca.nodes.Root
Pnfd	tosca.nodes.nfv.PNF	tosca.nodes.Root
Vnfd	tosca.nodes.nfv.VNF	tosca.nodes.Root

7.2 Data Types

7.2.1 toska.datatype.nfv.ServiceAvailability

7.2.1.1 Description

ServiceAvailability describes the information on the Service Availability Level for a particular virtual link flavour, as defined in ETSI GS NFV-IFA 014 [2] and ETSI GS NFV-REL 001 [i.3]. Table 7.2.1.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.1.1-1: Type name, shorthand, and URI

Shorthand Name	ServiceAvailability
Type Qualified Name	toscanfv:ServiceAvailability
Type URI	tosca.datatypes.nfv.ServiceAvailability

7.2.1.2 Properties

The properties of the ServiceAvailability data type shall comply with the provisions set out in table 7.2.1.2-1.

Table 7.2.1.2-1: Properties

Name	Required	Type	Constraints	Description
level	yes	string	valid_values: [level1, level2, level3]	Service available level as defined in [i.3]

7.2.1.3 Definition

The syntax of the ServiceAvailability data type shall comply with the following definition:

```
tosca.datatypes.nfv.ServiceAvailability:
  derived_from: toska.datatypes.Root
  description: service availability
  properties:
    level:
      type: string
      description: service availability levels
      required: true
      constraints:
        - valid_values: [ level1, level2, level3 ]
```

7.2.1.4 Examples

None.

7.2.2 toska.datatypes.nfv.VnfProfile

7.2.2.1 Description

The VnfProfile data type is defined in clause 6.2.52 of the present document.

7.2.3 toska.datatype.nfv.NsVIPProfile

7.2.3.1 Description

The NsVIPProfile data type describes additional instantiation data for a given NsVirtualLink used in a specific NS deployment flavour. Table 7.2.3.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.3.1-1: Type name, shorthand, and URI

Shorthand Name	NsVIPProfile
Type Qualified Name	toscanfv:NsVIPProfile
Type URI	tosca.datatypes.nfv.NsVIPProfile

7.2.3.2 Properties

The properties of the NsVIPProfile data type shall comply with the provisions set out in table 7.2.3.2-1.

Table 7.2.3.2-1: Properties

Name	Required	Type	Constraints	Description
max_bitrate_requirements	yes	tosca.datatypes.nfv.LinkBitrateRequirements		Specifies the maximum bitrate requirements for a VL instantiated according to this profile.
min_bitrate_requirements	yes	tosca.datatypes.nfv.LinkBitrateRequirements		Specifies the minimum bitrate requirements for a VL instantiated according to this profile.
qos	no	tosca.datatypes.nfv.NsVirtualLinkQos		Specifies the QoS requirements of a VL instantiated according to this profile.
service_availability	no	tosca.datatypes.nfv.ServiceAvailability		Network service virtual link service availability levels, as described in ETSI GS NFV-REL 001 [i.3].
NOTE:	A virtualLinkDescId property, which exists in ETSI GS NFV-IFA 014 [2] is not needed, as the NsVIPProfile is contained in the NsVirtualLink node.			

7.2.3.3 Definition

The syntax of the NsVIPProfile data type shall comply with the following definition:

```
tosca.datatypes.nfv.NsVIPProfile:
  derived_from: toska.datatypes.Root
  description: Describes additional instantiation data for a given NsVirtualLink used
in a specific NS deployment flavour.
  properties:
    max_bitrate_requirements:
      type: toska.datatypes.nfv.LinkBitrateRequirements
      description: Specifies the maximum bitrate requirements for a VL instantiated
according to this profile.
```

```

required: true
min_bitrate_requirements:
  type: toska.datatypes.nfv.LinkBitrateRequirements
  description: Specifies the minimum bitrate requirements for a VL instantiated
according to this profile.
  required: true
qos:
  type: toska.datatypes.nfv.NsVirtualLinkQos
  description: Specifies the QoS requirements of a VL instantiated according to this
profile.
  required: false
service_availability:
  type: toska.datatypes.nfv.ServiceAvailability
  description: Network service virtual link service availability levels, as
described in ETSI GS NFV-REL 001
  required: false

```

7.2.3.4 Examples

None.

7.2.3.5 Additional Requirements

None.

7.2.4 toska.datatypes.nfv.ConnectivityType

7.2.4.1 Description

The ConnectivityType data type is defined in clause 6.2.5 of the present document.

7.2.5 toska.datatypes.nfv.NsVirtualLinkQos

7.2.5.1 Description

The NsVirtualLinkQoS describes QoS data type a given NsVirtualLink used in an NS deployment flavour. Table 7.2.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.5.1-1: Type name, shorthand, and URI

Shorthand Name	NsVirtualLinkQos
Type Qualified Name	toscanfv:NsVirtualLinkQos
Type URI	tosca.datatypes.nfv.NsVirtualLinkQos

7.2.5.2 Properties

The properties of the NsVirtualLinkQos data type shall comply with the provisions set out in table 7.2.5.2-1.

Table 7.2.5.2-1: Properties

Name	Required	Type	Constraints	Description
priority	no	integer		Specifies the priority level in case of congestion on the underlying physical links.

7.2.5.3 Definition

The syntax of the `NsVirtualLinkQos` data type shall comply with the following definition:

```
tosca.datatypes.nfv.NsVirtualLinkQos:  
  derived_from: toska.datatypes.nfv.Qos  
  description: describes QoS data for a given VL used in a VNF deployment flavour  
  properties:  
    priority:  
      type: integer  
      description: Specifies the priority level in case of congestion on the underlying  
physical links  
      required: false
```

7.2.5.4 Examples

None.

7.2.5.5 Additional Requirements

None.

7.2.6 `tosca.datatypes.nfv.LinkBitrateRequirements`

7.2.6.1 Description

The `LinkBitrateRequirements` data type is defined in clause 6.2.29 of the present document.

7.2.7 `tosca.datatypes.nfv.VirtualLinkProtocolData`

7.4.7.1 Description

The `VirtualLinkProtocolData` data type is defined in clause 6.2.14 of the present document.

7.2.8 `tosca.datatypes.nfv.L2ProtocolData`

7.2.8.1 Description

The `L2ProtocolData` data type is defined in clause 6.2.15 of the present document.

7.2.9 `tosca.datatypes.nfv.L3ProtocolData`

7.2.9.1 Description

The `L3ProtocolData` data type is defined in clause 6.2.16 of the present document.

7.2.10 `tosca.datatypes.nfv.IpAllocationPool`

7.2.10.1 Description

The `IpAllocationPool` data type is defined in clause 6.2.17 of the present document.

7.2.11 `tosca.datatypes.nfv.CpProtocolData`

7.2.11.1 Description

The `CpProtocolData` data type is defined in clause 6.2.36 of the present document.

7.2.12 `tosca.datatypes.nfv.AddressData`

7.2.12.1 Description

The `AddressData` data type is defined in clause 6.2.3 of the present document.

7.2.13 `tosca.datatypes.nfv.L2AddressData`

7.2.13.1 Description

The `L2AddressData` data type is defined in clause 6.2.1 of the present document.

7.2.14 `tosca.datatypes.nfv.L3AddressData`

7.2.14.1 Description

The `L3AddressData` data type is defined in clause 6.2.2 of the present document.

7.2.15 `tosca.datatypes.nfv.Qos`

7.2.15.1 Description

The `Qos` data type is defined in clause 6.2.30 of the present document.

7.3 Artifact Types

None.

7.4 Capabilities Types

7.4.1 `tosca.capabilities.nfv.VirtualLinkable`

7.4.1.1 Description

The `VirtualLinkable` capability type is defined in clause 6.4.2 of the present document.

7.5 Requirements Types

None.

7.6 Relationship Types

7.6.1 `tosca.relationships.nfv.VirtualLinksTo`

7.6.1.1 Description

The `VirtualLinksTo` relationship type is defined in clause 6.6.2 of the present document representing an association relationship between the VNF or PNF or Sap of a Nested NS and `NsVirtualLink` node types when used in a NSD.

7.7 Interface Types

7.7.1 `tosca.interfaces.nfv.Nslcm`

7.7.1.1 Description

The `tosca.interfaces.nfv.Nslcm` interface type contains a set of TOSCA operations corresponding to the following NS LCM operations defined in ETSI GS NFV-IFA 013 [i.8]:

- Instantiate NS
- Scale NS
- Update NS
- Heal NS
- Terminate NS

The interface also contains TOSCA operations corresponding to preamble and postamble to the execution of the aforementioned base operations. The name of these operations is constructed according to the following pattern:

`<base_operation_name>_start` for a preamble

`<base_operation_name>_end` for a postamble

The designations ("`_start`", "`_end`") in the name of TOSCA operations are postfixes so that related operations are adjacent in an alphabetical listing.

Table 7.7.1.1-1 specifies the declared names for this interface type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.7.1.1-1: Type name, shorthand, and URI

Shorthand Name	<code>Nslcm</code>
Type Qualified Name	<code>toscanfv:Nslcm</code>
Type URI	<code>tosca.interfaces.nfv.Nslcm</code>

7.7.1.2 Definition

The syntax of the `Nslcm` interface type shall comply with the following definition:

```
tosca.interfaces.nfv.Nslcm:
  derived_from: toasca.interfaces.Root
  description: This interface encompasses a set of TOSCA operations corresponding to
  NS LCM operations defined in ETSI GS NFV-IFA 013. as well as to preamble and postamble
  procedures to the execution of the NS LCM operations.
  instantiate_start:
    description: Preamble to execution of the instantiate operation
  instantiate:
```



```

description: Base procedure for instantiating an NS, corresponding to the
Instantiate NS operation defined in ETSI GS NFV-IFA 013.
instantiate_end:
description: Postamble to the execution of the instantiate operation
terminate_start:
description: Preamble to execution of the terminate operation
terminate:
description: Base procedure for terminating an NS, corresponding to the Terminate
NS operation defined in ETSI GS NFV-IFA 013.
terminate_end:
description: Postamble to the execution of the terminate operation
update_start:
description: Preamble to execution of the update operation
update:
description: Base procedure for updating an NS, corresponding to the Update NS
operation defined in ETSI GS NFV-IFA 013.
update_end:
description: Postamble to the execution of the update operation
scale_start:
description: Preamble to execution of the scale operation
scale:
description: Base procedure for scaling an NS, corresponding to the Scale NS
operation defined in ETSI GS NFV-IFA 013.
scale_end:
description: Postamble to the execution of the scale operation
heal_start:
description: Preamble to execution of the heal operation
heal:
description: Base procedure for healing an NS, corresponding to the Heal NS
operation defined in ETSI GS NFV-IFA 013.
heal_end:
description: Postamble to the execution of the heal operation

```

7.7.1.3 Additional Requirements

The implementation and inputs keynames specified in TOSCA-Simple-Profile-YAML-v1.2 [3] for an operation definition may be included for each operation listed in the Nslcm interface definition.

When a TOSCA operation representing an NS LCM operation does not have an associated implementation keyname, the default implementation provided by the NFVO for this NS LCM operation applies.

The NSD consumer shall make available all parameters from the message invoking the NS LCM operation as inputs to the corresponding TOSCA interface operations. The `inputs` keyname can be used to specify additional input parameters for executing the operation.

The implementation of preamble and postamble TOSCA operations (e.g. `instantiate_start`), if present, is invoked with the same parameters as the corresponding base TOSCA operation (e.g. `instantiate`). The inputs of the preamble and postamble operations shall not be defined in the NSD.

If an implementation is associated to a TOSCA operation that represents a preamble or a postamble to a VNF LCM operation, the implementation logic is executed before or after the execution of the VNF LCM operation implementation, respectively.

7.7.1.4 Support of LCM scripts

In ETSI GS NFV-IFA 014 [2], the definition of the "LifeCycleManagementScript" information element of the NSD associates LCM scripts with events, where an event can be an external or an internal stimulus. These events are mapped to TOSCA operations of the NS node type in the following way:

- external stimuli are mapped to TOSCA operations corresponding to the NS LCM operations defined in ETSI GS NFV-IFA 013 [i.8];
- internal stimuli are mapped to preamble and postamble of these TOSCA operations.

LCM scripts can be regarded as artifacts that provide an NS-specific implementation of the TOSCA operation corresponding to the stimulus.

The script input parameters shall be provided to the script according to the declaration in the inputs field of the operation definition. The artifact type definition shall enable identifying the DSL used by the script. The artifact type definition for Python is provided in section 5.4.4.1 of TOSCA-Simple-Profile-YAML-v1.2 [3]. The artifact definition for Mistral is provided in clause A.7.2 of the present document.

7.7.1.5 Examples

The following example template fragments illustrate the concept. An LCM script is associated with the `instantiate_end` operation. As no LCM script is associated to the `instantiate` operation, its default implementation runs and before running the `post-instantiate-script`. The inputs section of the `instantiate_end` operation definition provides additional input values to the `post-instantiate-script`, and the TOSCA artifacts definition conveys the type of DSL used as a scripting language.

```
tosca_definitions_version: tosca_simple_yaml_1_2

imports:
- ..

node_types:
  MyCompany.SunshineVPN.1_0.1_0:
    derived_from: tosca.nodes.nfv.NS
    ..

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineVPN.1_0.1_0
    ..

node_templates:
  SunshineVPN:
    type: MyCompany.SunshineVPN.1_0.1_0
    ..
  interfaces:
    Nslcm:
      instantiate_end:
        implementation: post-instantiate-script
        inputs:
          script_input_1: value_1
          script_input_2: value_2

artifacts:
  post-instantiate-script:
    description: Instantiate workflow script
    type: tosca.artifacts.Implementation.Python
    file: instantiate.py
    #repository: ..
    #deploy_path: ..
    ..
```

7.8 Node Types

7.8.1 toska.nodes.nfv.NS

7.8.1.1 Description

The NFV Network Service (NS) node type describes a NS in terms of deployment, operational behaviour, and requirements, as defined by ETSI GS NFV-IFA 014 [2]. Table 7.8.1.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.8.1.1-1: Type name, shorthand, and URI

Shorthand Name	NS
Type Qualified Name	toscanfv:NS
Type URI	tosca.nodes.nfv.NS

7.8.1.2 Properties

The properties of the NS node type shall comply with the provisions set out in table 7.8.1.2-1.

Table 7.8.1.2-1: Properties

Name	Required	Type	Constraints	Description
descriptor_id	yes	string		Identifier of this NS descriptor.
designer	yes	string		Identifies the designer of the NSD.
version	yes	string		Identifies the version of the NSD.
name	yes	string		Provides the human readable name of the NSD.
invariant_id	yes	string		Identifies an NSD in a version independent manner. This attribute is invariant across versions of NSD.

The value of the descriptor_id string shall comply with an UUID format as specified in section 3 of [9].

7.8.1.3 Attributes

None.

7.8.1.4 Requirements

The requirements of the NS node type shall comply with the provisions set out in table 7.8.1.4-1.

Table 7.8.1.4-1: Requirements

Name	Required	Type	Constraints	Description
virtual_link	yes	tosca.nodes.nfv.Ns VirtualLink		Describes the requirements for linking to virtual link

7.8.1.5 Capabilities

None.

7.8.1.6 Definition

The syntax of the NS node type shall comply with the following definition:

```

tosca.nodes.nfv.NS:
  derived_from: toska.nodes.Root
  properties:
    descriptor_id:
      type: string # GUID
      description: Identifier of this NS descriptor
      required: true
    designer:
      type: string
      description: Identifies the designer of the NSD.
      required: true
    version:
      type: string
      description: Identifies the version of the NSD.
      required: true
    name:
      type: string
      description: Provides the human readable name of the NSD.
      required: true
    invariant_id:
      type: string
      description: Identifies an NSD in a version independent manner. This
attribute is invariant across versions of NSD
      required: true
  requirements:
    - virtual_link:
      capability: toska.capabilities.nfv.VirtualLinkable
      relationship: toska.relationships.nfv.VirtualLinksTo
      node: toska.nodes.nfv.NsVirtualLink
      occurrences: [ 0, 1 ]
  interfaces:
    Nslcm:
      type: toska.interfaces.nfv.Nslcm

```

7.8.1.7 Artifact

None.

7.8.1.8 Additional requirements

For a given NSD, a new VNF node type shall be defined following the below requirements:

- a) The node type shall be derived from: toska.nodes.nfv.NS.
- b) All properties listed in toska.nodes.nfv.NS where the "required:" field is set to "true" shall be included with their values indicated as constraints.
- c) Properties listed in in toska.nodes.nfv.NS where the "required:" field is set to "false" may be included.
- d) The capabilities, requirements, interfaces of toska.nodes.nfv.VNF shall be preserved.
- e) Depending on the number of SAPs of the NS, additional requirements for VirtualLinkable capability shall be defined. In this case, it is the NSD author's choice to use the requirement for VirtualLinkable capability defined in the toska.nodes.nfv.NS node type or use only the additional requirements defined in the derived NS specific node type. In the latter case, the virtual_link requirement should be included in the node type definition with occurrences [0, 0].

7.8.2 toska.nodes.nfv.Sap

7.8.2.1 Description

The Service Access Point (SAP) node type describes a connection point where a NS can be accessed, as defined by ETSI GS NFV-IFA 014 [2]. Table 7.8.2.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.8.2.1-1: Type name, shorthand, and URI

Shorthand Name	Sap
Type Qualified Name	toscanfv:Sap
Type URI	tosca.nodes.nfv.Sap

7.8.2.2 Properties

The properties applied to Sap node are derived from Cp node type as defined in clause 6.8.7 of the present document.

7.8.2.3 Attributes

None.

7.8.2.4 Requirements

The requirements of the Sap node type shall comply with the provisions set out in table 7.8.2.4-1.

Table 7.8.2.4-1: Requirements

Name	Required	Type	Constraints	Description
external_virtual_link	no	tosca.capabilities.nfv.VirtualLinkable		Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability
internal_virtual_link	yes	tosca.capabilities.nfv.VirtualLinkable		Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability

7.8.2.5 Capabilities

None.

7.8.2.6 Definition

The syntax of the Sap node type shall comply with the following definition:

```
tosca.nodes.nfv.Sap:
  derived_from: toska.nodes.nfv.Cp
  description: node definition of SAP.

  requirements:
  - external_virtual_link:
    capability: toska.capabilities.nfv.VirtualLinkable
    relationship: toska.relationships.nfv.VirtualLinksTo
  - internal_virtual_link:
    capability: toska.capabilities.nfv.VirtualLinkable
    relationship: toska.relationships.nfv.VirtualLinksTo
```

7.8.2.7 Additional requirements

A node template of this type is used to represent a SAP only in the case the Sap is connected to a NsVirtualLink inside a NSD. The node template has the following requirements:

- `internal_virtual_link` requirement to allow to connect it to a NsVirtualLink inside a NSD
- `external_virtual_link` requirement to allow to connect it to a NsVirtualLink outside a NSD

In the case where a Sap is exposed by a VNF external connection point, a PNF external connection point or a Sap of the nested NS, the Sap node type does not apply.

7.8.2.8 Example

In a typical scenario, the Sap node template will be part of a service template representing a certain NS deployment flavour. The service template substitutes for an abstract NS node type. In this substitution, the `virtual_link` requirement of the abstract NS node type is mapped to the `external_virtual_link` requirement of the Sap node. This example is illustrated in clause A.7.3.

When a Sap re-exposes a VNF external connection point, the service template does not require an explicit node template of type Sap in a typical scenario where an abstract NS node type is substituted by a service template representing a certain NS deployment flavour. In this substitution, the `virtual_link` requirement of the abstract NS node type is mapped to the `virtual_link` requirement of the VNF node. This example is illustrated in clause A.7.2.

7.8.3 `tosca.nodes.nfv.NsVirtualLink`

7.8.3.1 Description

The NsVirtualLink node type represents the NsVirtualLinkDesc information element as defined in ETSI GS NFV-IFA 014 [2], which describes the requirements for a virtual link of a network service. Table 7.8.3.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.8.3.1-1: Type name, shorthand, and URI

Shorthand Name	NsVirtualLink
Type Qualified Name	toscanfv: NsVirtualLink
Type URI	tosca.nodes.nfv.NsVirtualLink

7.8.3.2 Properties

The properties of the NsVirtualLink node type shall comply with the provisions set out in table 7.8.3.2-1.

Table 7.8.3.2-1: Properties

Name	Required	Type	Constraints	Description
<code>vl_profile</code>	yes	<code>tosca.datatype.nfv.NsVIPProfile</code>		Specifies instantiation parameters for a virtual link of a particular NS deployment flavour.
<code>connectivity_type</code>	yes	<code>tosca.datatypes.nfv.ConnectivityType</code>		Network service virtual link connectivity type.
<code>test_access</code>	no	list of string	Valid values: <code>passive_monitoring</code> , <code>active_loopback</code>	Test access facilities available on the VL.
<code>description</code>	no	string		Human readable information on the purpose of the virtual link (e.g. VL for control plane traffic).

7.8.3.3 Attributes

None.

7.8.3.4 Requirements

None.

7.8.3.5 Capabilities

The capabilities of the NsVirtualLink node type shall comply with the provisions set out in table 7.8.3.5-1.

Table 7.8.3.5-1: Capabilities

Name	Type	Constraints	Description
virtual_linkable	tosca.capabilities.nfv.VirtualLinkable		VirtualLinkable capability

7.8.3.6 Definition

The syntax of the NsVirtualLink node type shall comply with the following definition:

```

tosca.nodes.nfv.NsVirtualLink:
  derived_from: toska.nodes.Root
  description: node definition of Virtual Links
  properties:
    type: toska.datatypes.nfv.NsVlProfile # only covers min/max bitrate
  requirements:
    description: Specifies instantiation parameters for a virtual link of a
    particular NS deployment flavour.
    required: true
    connectivity_type:
      type: toska.datatypes.nfv.ConnectivityType
      required: true
    test_access:
      type: list
      description: Test access facilities available on the VL
      required: false
    entry_schema:
      type: string
      constraints:
        - valid_values: [ passive_monitoring, active_loopback ]
      description:
        type: string
        required: false
        description: Human readable information on the purpose of the virtual link
        (e.g. VL for control plane traffic).
    capabilities:
      virtual_linkable:
        type: toska.capabilities.nfv.VirtualLinkable

```

7.8.3.7 Artifact

None.

7.8.3.8 Additional Requirements

None.

7.8.3.9 Example

None.

7.8.4 `tosca.nodes.nfv.Cp`

7.8.4.1 Description

The Cp node type is defined in clause 6.8.7 of the present document.

7.9 Group Types

7.9.1 `tosca.groups.nfv.PlacementGroup`

7.9.1.1 Description

The PlacementGroup group type is defined in clause 6.9.1 of the present document describing the affinity or anti-affinity relationship applicable between VNF instances created using different VNFDs, the Virtual Link instances created using different VLDs or the nested NS instances created using different NSDs when used in a NSD.

7.10 Policy Types

7.10.1 AffinityRule, AntiAffinityRule

7.10.1.1 Description

The AffinityRule and AntiAffinityRule policy types as defined in clause 6.10.10 when used in a NSD TOSCA service template specify the affinity or anti-affinity rules applicable:

- If there is only one node template with node type `tosca.nodes.nfv.VNF` or `tosca.nodes.nfv.NsVirtualLink` set as the targets, the AffinityRule or AntiAffinityRule applies between the instances to be created based on the same VNFD, or between VLs to be created based on the same VLD as described in ETSI GS NFV-IFA 014 [2].
- If there are more than one node templates with node type `tosca.nodes.nfv.VNF` or `tosca.nodes.nfv.NsVirtualLink` or `tosca.nodes.nfv.NS` set as the targets, or a group with type `tosca.groups.nfv.PlacementGroup` which contains more than one members set as targets, the AffinityRule or AntiAffinityRule applies between VNF instances created using different VNFD, the Virtual Link instances created using different VLD or the nested NS instances created using different NSD. as described in ETSI GS NFV-IFA 014 [2].

7.10.1.2 Targets

The targets of the AffinityRule and AntiAffinityRule policy types shall comply with the provisions set out in table 7.10.1.2-1 when used in a NSD.

Table 7.10.1.2-1: Targets

Name	Required	Type	Constraints	Description
targets	Yes	string[]	Valid types: <code>tosca.nodes.nfv.VNF</code> , <code>tosca.nodes.nfv.NsVirtualLink</code> , <code>tosca.nodes.nfv.NS</code> , <code>tosca.groups.nfv.PlacementGroup</code>	In case of LocalAffinityOrAntiAffinityRule as defined in ETSI GS NFV-IFA 014 [2], the valid type of the targets is <code>tosca.nodes.nfv.VNF</code> or <code>tosca.nodes.nfv.NsVirtualLink</code> . In case of affinityOrAntiAffinityGroup as defined in ETSI GS NFV-IFA 014 [2], the valid types of the targets are: <code>tosca.nodes.nfv.VNF</code> , <code>tosca.nodes.nfv.NsVirtualLink</code> and <code>tosca.nodes.nfv.NS</code> or a <code>tosca.groups.nfv.PlacementGroup</code> .

7.10.1.3 Examples

The following example template fragments illustrate the concepts:

```
node_templates:
  VNF_1:
    type: toasca.nodes.nfv.exampleVNF

policies:
  policy_affinity_local_VNF_1:
    type: toasca.policies.nfv.AffinityRule
    targets: [ VNF_1 ]
    properties:
      scope: nfvi_node
```

The above example illustrates a local affinity rule for all the instances of VNF_1.

```
node_template:
  VNF_1:
    type: toasca.nodes.nfv.Vdu.exampleVNF_1

  VNF_2:
    type: toasca.nodes.nfv.Vdu.exampleVNF_2

groups:
  affinityOrAntiAffinityGroup_1:
    type: toasca.groups.nfv.PlacementGroup
    members: [ VNF_1, VNF_2 ]

policies:
  policy_antiaffinity_group_1:
    type: toasca.policies.nfv.AntiAffinityRule
    targets: [ affinityOrAntiAffinityGroup_1 ]
    properties:
      scope: nfvi_node
```

The above example illustrates an anti-affinity policy among a group which contains VNF_1 and VNF_2 as members.

7.10.2 toasca.policies.nfv.SecurityGroupRule

7.10.2.1 Description

The SecurityGroupRule policy type as defined in clause 6.10.11 of the present document when used in a NSD specifies the matching criteria for the ingress and/or egress traffic to and from visited SAPs.

7.10.2.2 Additional Requirements

The valid type of the targets of the SecurityGroupRule policy when used in a NSD shall set to toasca.nodes.nfv.Sap.

8 PNF-D TOSCA model

8.1 Introduction

The PNF-D information model specified by ETSI GS NFV-IFA 014 [2] is mapped to the TOSCA concepts. It represents as TOSCA topology template to be used by NFVO for preparing network connection.

Table 8.1-1 shows the TOSCA Type "derived from" values used when applying the TOSCA Simple YAML Profile [3] to the PNF-D.

Table 8.1-1: Mapping of ETSI GS NFV-IFA 014 [2] information elements with TOSCA types

ETSI NFV Information Element ETSI GS NFV-IFA 014 [2]	TOSCA type	Derived from
Pnf	tosca.nodes.nfv.PNF	tosca.nodes.Root
PnfExtCpd (PNF External Connection Point)	tosca.nodes.nfv.PnfExtCp	tosca.nodes.Root

Figure 8.1-1 provides an overview of the TOSCA node types used to build a service template for a PNF-D.

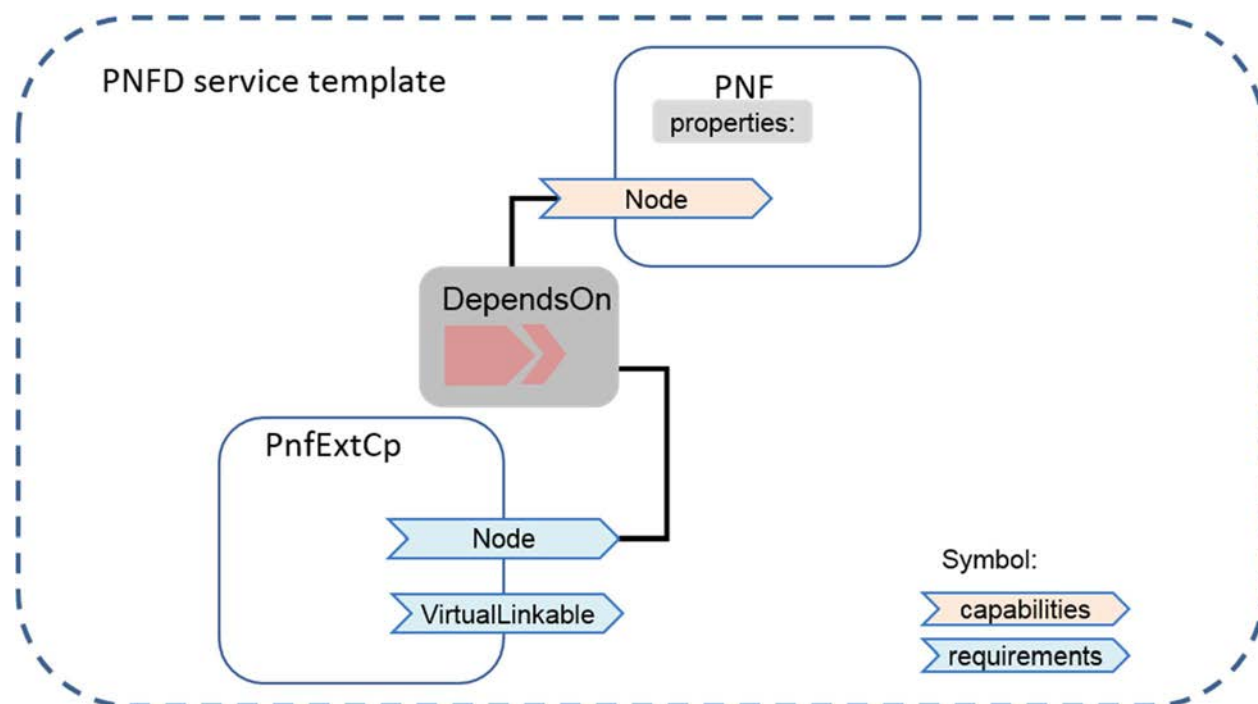


Figure 8.1-1: Service template PNF-D overview

8.2 Data Types

8.2.1 `tosca.datatypes.nfv.CpProtocolData`

8.2.1.1 Description

The `CpProtocolData` data type is defined in clause 6.2.36 of the present document.

8.2.2 `tosca.datatypes.nfv.AddressData`

8.2.2.1 Description

The `AddressData` data type is defined in clause 6.2.3 of the present document.

8.2.3 `tosca.datatypes.nfv.L2AddressData`

8.2.3.1 Description

The `L2AddressData` data type is defined in clause 6.2.1 of the present document.

8.2.4 `tosca.datatypes.nfv.L3AddressData`

8.2.4.1 Description

The `L3AddressData` data type is defined in clause 6.2.2 of the present document.

8.2.5 `tosca.datatypes.nfv.LocationInfo`

8.2.5.1 Description

The `LocationInfo` data type represents geographical information on the location where a PNF is deployed as specified in ETSI GS NFV-IFA 011 [1]. Table 8.2.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.2.5.1-1: Type name, shorthand, and URI

Shorthand Name	<code>LocationInfo</code>
Type Qualified Name	<code>toscanfv:LocationInfo</code>
Type URI	<code>tosca.datatypes.nfv.LocationInfo</code>

8.2.5.2 Properties

The properties of the `LocationInfo` data type shall comply with the provisions set out in table 8.2.5.2-1.

Table 8.2.5.2-1: Properties

Name	Required	Type	Constraints	Description
<code>country_code</code>	yes	string		Shall be a two-letter ISO 3166 [10] country code in capital letters.
<code>civic_address_element</code>	no	list of <code>tosca.datatypes.nfv.CivicAddressElement</code>		Elements composing the civic address where the PNF is deployed.

8.2.5.3 Definition

The syntax of the `LocationInfo` data type shall comply with the following definition:

```
tosca.datatypes.nfv.LocationInfo:
  derived_from: toasca.datatypes.Root
  description: Represents geographical information on the location where a PNF is
  deployed.
  properties:
    country_code:
      type: string #two-letter ISO 3166 country code
```

```

description: Country code
required: true
civic_address_element:
  type: list
  entry-schema
    type: toasca.datatypes.nfv.CivicAddressElement
description: Elements composing the civic address where the PNF is deployed.
required: false

```

8.2.5.4 Examples

```

<some_tosca_entity>:
  properties:
    geographical_location_info:
      country_code: FR
      civic_address_element:
        - element_1
          ca_type: 3
          ca_value: Paris

```

8.2.5.5 Additional Requirements

None.

8.2.6 toasca.datatypes.nfv.CivicAddressElement

8.2.6.1 Description

The CivicAddressElement data type represents an element of a civic location as specified in IETF RFC 4776 [11]. Table 8.2.6.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.2.6.1-1: Type name, shorthand, and URI

Shorthand Name	CivicAddressElement
Type Qualified Name	toscanfv: CivicAddressElement
Type URI	tosca.datatypes.nfv. CivicAddressElement

8.2.6.2 Properties

The properties of the CivicAddressElement data type shall comply with the provisions set out in table 8.2.6.2-1.

Table 8.2.6.2-1

Name	Required	Type	Constraints	Description
ca_type	yes	string		Describe the content type of caValue. The value of caType shall comply with section 3.4 of IETF RFC 4776 [11].
ca_value	yes	string		Content of civic address element corresponding to the caType. The format caValue shall comply with section 3.4 of IETF RFC 4776 [11].

8.2.6.3 Definition

The syntax of the CivicAddressElement data type shall comply with the following definition:

```
tosca.datatypes.nfv.CivicAddressElement:
  derived_from: toska.datatypes.Root
  description: Represents an element of a civic location as specified in IETF
  RFC 4776 [11].
  properties:
    ca_type:
      type: string # RFC4776
      description: caType as per RFC4776
      required: true
    ca_value:
      type: string # RFC4776
      description: caValue as per RFC4776.
      required: true
```

8.2.6.4 Examples

See clause 8.2.5.4.

8.2.6.5 Additional Requirements

None.

8.3 Artifact Types

None.

8.4 Capabilities Types

8.4.1 toska.capabilities.nfv.VirtualLinkable

8.4.1.1 Description

The VirtualLinkable capability type is defined in clause 6.4.2 of the present document.

8.5 Requirements Types

None.

8.6 Relationship Types

8.6.1 toska.relationships.nfv.VirtualLinksTo

8.6.1.1 Description

The VirtualLinksTo relationship type is defined in clause 6.6.2 of the present document representing an association relationship between a PNF external connection point and an NsVirtualLink node type.

8.7 Interface Types

None.

8.8 Node Types

8.8.1 `tosca.nodes.nfv.PNF`

8.8.1.1 Description

The Physical Network Function (PNF) node type describes a PNF in terms of deployment behaviour requirements, which it contains PNFD identifier, version and functional description and so on as defined by ETSI GS NFV-IFA 014 [2]. Table 8.8.1.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.8.1.1-1: Type name, shorthand, and URI

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

8.8.1.2 Properties

The properties of the PNF node type shall comply with the provisions set out in table 8.8.1.2-1.

Table 8.8.1.2-1: Properties

Name	Required	Type	Constraints	Description
descriptor_id	yes	string		Identifier of this PNFD information element. It uniquely identifies the PNFD.
function_description	yes	string		Describes the PNF function.
provider	yes	string		Identifies the provider of the PNFD.
version	yes	string		Identifies the version of the PNFD.
descriptor_invariant_id	yes	string		Identifier of this PNFD in a version independent manner. This attribute is invariant across versions of PNFD.
name	yes	string		Name to identify the PNFD.
geographical_location_info	no	tosca.data type.nfv.L ocationInf o		Provides information about the geographical location (e.g. geographic coordinates or address of the building, ...) of the PNF.

The value of the `descriptor_id` string and the value of the `descriptor_invariant_id` string shall comply with an UUID format as specified in section 3 of [8].

8.8.1.3 Attributes

None.

8.8.1.4 Requirements

The requirements of the VNF node type shall comply with the provisions set out in table 8.8.1.4-1.

Table 8.8.1.4-1: Requirements

Name	Required	Type	Constraints	Description
virtual_link	yes	tosca.nodes.nfv.Vnf VirtualLink		Describes the requirements for linking to virtual link

8.8.1.5 Capabilities

None.

8.8.1.6 Definition

```

tosca.nodes.nfv.PNF:
  derived_from: tosca.nodes.Root
  properties:
    descriptor_id: # instead of pnf_id
      type: string # GUID
      required: true
    function_description:
      type: string
      required: true
    provider:
      type: string
      required: true
    version:
      type: string
      required: true
    descriptor_invariant_id: # instead of pnf-invariant-id
      type: string # GUID
      required: true
    name:
      type: string
      required: true
    geographical_location_info:
      type: tosca.datatypes.nfv.LocationInfo
      required: false
    requirements:
      - virtual_link:
          capability: tosca.capabilities.nfv.VirtualLinkable
          relationship: tosca.relationships.nfv.VirtualLinksTo
          occurrences: [ 0, 1 ]
  # Additional requirements shall be defined in the PNF specific node type
  (deriving from tosca.nodes.nfv.PNF) corresponding to NS virtual links that need to
  connect to PnfExtCps

```

8.8.1.7 Artifact

None.

8.8.1.8 Additional Requirements

For a given PNFD, a new PNF node type shall be defined following the below requirements:

- a) The node type shall be derived from: tosca.nodes.nfv.PNF.
- b) All properties listed in tosca.nodes.nfv.PNF where the "required:" field is set to "true" shall be included with their values indicates as constraints.

- c) The requirements of `tosca.nodes.nfv.PNF` shall be preserved.
- d) Depending on the number of external connection points of the PNF that need to connect to NS virtual links, additional requirements for `VirtualLinkable` capability shall be defined. In this case, it is the PNF author's choice to use the requirement for `VirtualLinkable` capability defined in the `tosca.nodes.nfv.PNF` node type or use only the additional requirements defined in the derived PNF specific node type.

8.8.1.9 Example

See clause A.10.

8.8.2 `tosca.nodes.nfv.PnfExtCp`

8.8.2.1 Description

The `PnfExtCp` node type describes the characteristics of an external interface, a.k.a. an external CP, where to connect the PNF to a VL, as defined by ETSI GS NFV-IFA 014 [2]. Table 8.8.2.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.8.2.1-1: Type name, shorthand, and URI

Shorthand Name	<code>PnfExtCp</code>
Type Qualified Name	<code>toscanfv:PnfExtCp</code>
Type URI	<code>tosca.nodes.nfv.PnfExtCp</code>

8.8.2.2 Properties

The properties applied to `PnfExtCp` node are derived from `Cp` node type.

8.8.2.3 Attributes

None.

8.8.2.4 Requirements

The requirements of the `PnfExtCp` node type shall comply with the provisions set out in table 8.8.2.4-1.

Table 8.8.2.4-1: Requirements

Name	Required	Type	Constraints	Description
<code>external_virtual_link</code>	no	<code>tosca.capabilities.nfv.VirtualLinkable</code>		Specifies that CP instances require to be connected to a node that has a <code>VirtualLinkable</code> capability

8.8.2.5 Capabilities

None.

8.8.2.6 Definition

The syntax of the `PnfExtCp` node type shall comply with the following definition:

```
tosca.nodes.nfv.PnfExtCp:
  derived_from: toasca.nodes.nfv.Cp
  description: node definition of PnfExtCp.
  requirements:
    - external_virtual_link:
```



```
capability: toasca.capabilities.nfv.VirtualLinkable  
relationship: toasca.relationships.nfv.VirtualLinksTo
```

8.8.3 toasca.nodes.nfv.Cp

8.8.3.1 Description

The Cp node type is defined in clause 6.8.7 of the present document.

8.9 Group Types

None.

8.10 Policy Types

8.10.1 toasca.policies.nfv.SecurityGroupRule

8.10.1.1 Description

The SecurityGroupRule policy type as defined in clause 6.10.11 of the present document when used in a PNFD specifies the matching criteria for the ingress and/or egress traffic to and from visited PNF external connection points.

8.10.1.2 Additional Requirements

The valid type of the targets of the SecurityGroupRule policy when used in a PNFD shall set to toasca.nodes.nfv.PnfExtCp.

Annex A (informative): Examples

A.1 Deployment flavour design mapping

A.1.1 Introduction

This annex describes the main design principle for VNF deployment flavour and the mapping between VNF deployment flavour elements to TOSCA types.

A.1.2 Design principle

Each Deployment flavour as specified in ETSI GS NFV-IFA 011 [1] describes a given deployment configuration of a VNF in terms of its internal topology and resource needs. Different deployment flavours can define different topologies of the same VNF, with different scalingAspect, different VDUs and different internal connectivity requirements. The idea of VNF deployment flavour as specified in [1] is that each deployment flavour describes the required vduProfiles and virtualLinkProfiles which are the additional instantiation data for the given VDUs and virtualLinks, once a specific deployment flavour has been chosen at the instantiation time, in order to successfully deploy the given VDUs and virtualLinks, the bindings between the VDU with the corresponding VduProfile and the virtualLinkDesc with the corresponding virtualLinkProfile are required.

To achieve the concept of deployment flavour by using TOSCA, the main design principle is to describe each deployment flavour as a standalone implementable TOSCA service template, and binding VDU and virtualLink with the corresponding VduProfile and virtualLinkProfile, respectively, together at the design time. Once a specific deployment flavour has been chosen at the instantiation time, the corresponding TOSCA service template will be used for deploying the VNF with the given deployment flavour.

Figure A.1.2-1 shows the general principle for a VNF deployment flavour design.

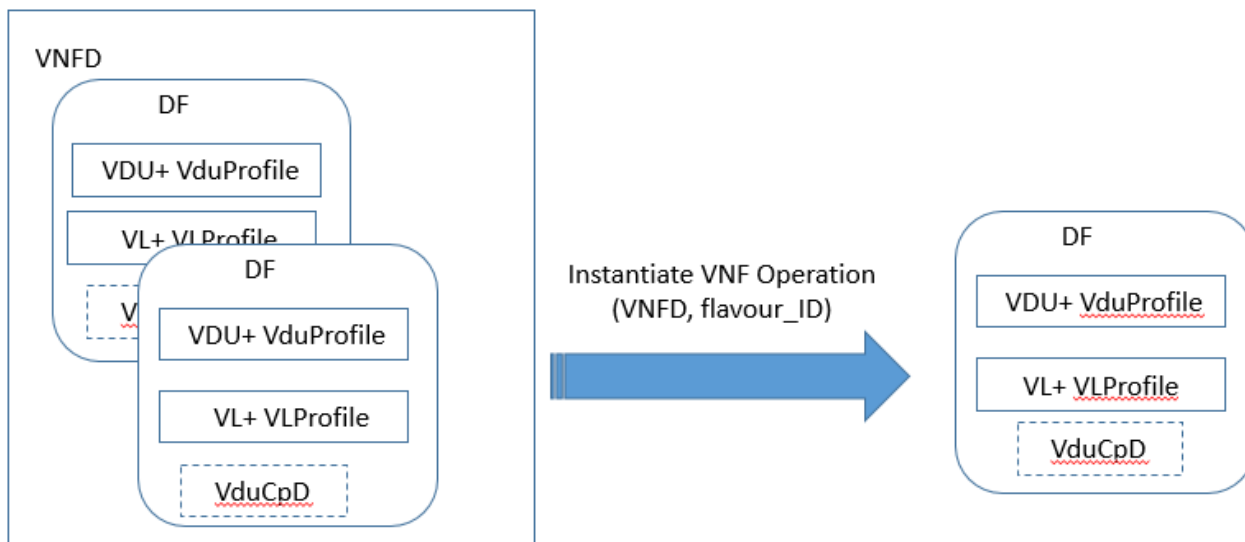


Figure A.1.2-1: General principle for a VNF deployment flavour design

A.2 VNFD with deployment flavour modelling design example

Deployment flavours are represented as deployable TOSCA topology templates. This way one VNF service template represents one deployment flavour, and different deployment flavours are described by different VNF service templates. This is in line with the idea that different deployment flavours can define different topologies of the same VNF, with different scaling aspects, different VDUs and different internal connectivity.

In order to represent a VNF a top-level service template is used. This top-level service template contains a topology template with only an abstract VNF node which defines the common parts of the different deployment flavours (such as product information, modifiable attributes and parts of the lifecycle management interface definition). It also sets a constraint on the deployment flavour property (the required value of the flavour_id property); this constraint comes from the VNF instantiation request which contains a flavour_id selected among those available in the VNFD.

As a result, the VNFM will look into the available further service templates representing the different VNF deployment flavours of the VNF and use the one that has the matching flavour_id property value to substitute for the abstract VNF. These are the low-level service templates.

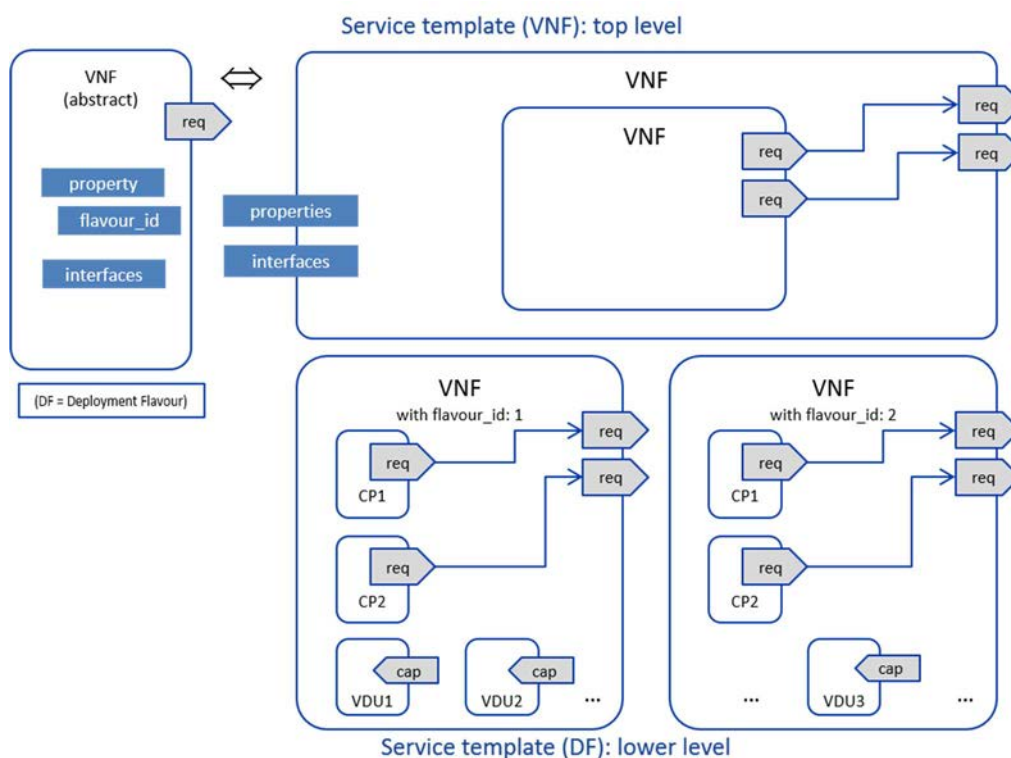


Figure A.2-1: VNFD overview with deployment flavour

A VNFD contains a TOSCA top-level Service Template as entry point in the VNF package and one or more TOSCA low-level Service templates representing the different deployment flavours (see figure A.2-1). The VNFD is interpreted by an NFVO or VNF manager. In this example, the templates describe two variants of the VNF each corresponding to a deployment flavour: a simple one and a complex one. The simple VNF consists of one server: a DB backend whereas, the complex VNF variant consists of minimum three DB backend servers and one serviceNode, which may be scaled out in one-size increments.

SunshineDB: VNFD-top level

sunshine.vnfd.tosca.yaml

```
tosca_definitions_version: tosca_simple_yaml_1_2
```

```
description: Relational database, non-scalable
```

```

imports:
- etsi_nfv_sol001_vnfd_2_5_1_types.yaml # all of TOSCA types as defined in NFV SOL
001 GS
- sunshineVNF.yaml # contains the VNF node type definition
- sunshinedbsimple.vnfd.tosca.yaml
- sunshinedbcomplex.vnfd.tosca.yaml

topology_template:
  inputs:
    selected_flavour:
      type: string
      description: VNF deployment flavour selected by the consumer. It is provided in
the API

  node_templates:
    SunshineDB:
      type: MyCompany.SunshineDB.1_0.1_0
      properties:
        flavour_id: { get_input: selected_flavour }
        descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
        provider: MyCompany
        product_name: SunshineDB
        software_version: '1.0'
        descriptor_version: '1.0'
        vnfm_info:
          - MyCompany
      requirements:
        #- virtual_link_backend # mapped in lower-level templates
        #- virtual_link_service # mapped in lower-level templates
        # get_input function would be used by a TOSCA orchestrator at run time to
access the selected flavour id. If the deployment is not done by a TOSCA orchestrator
above function may not be needed

```

The sunshineVNF.yaml file has the following content:

```

tosca_definitions_version: tosca_simple_yaml_1_2

description: Relational database, non-scalable

imports:
- etsi_nfv_sol001_vnfd_2_5_1_types.yaml # all of TOSCA types as defined in NFV SOL
001 GS

data_types:
  MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      parameter_1:
        type: string
        required: true
        default: value_1
      parameter_2:
        type: string
        required: false
        default: value_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:

```

```

descriptor_id:
  type: string
  constraints: [ valid_values: [ b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ] ]
  default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
provider:
  type: string
  constraints: [ valid_values: [ MyCompany ] ]
  default: MyCompany
product_name:
  type: string
  constraints: [ valid_values: [ SunshineDB ] ]
  default: SunshineDB
software_version:
  type: string
  constraints: [ valid_values: [ '1.0' ] ]
  default: '1.0'
descriptor_version:
  type: string
  constraints: [ valid_values: [ '1.0' ] ]
  default: '1.0'
flavour_id:
  type: string
  constraints: [ valid_values: [ simple, complex ] ]
  default: simple
flavour_description:
  type: string
  default: "" #empty string
vnfm_info:
  type: list
  entry_schema:
    type: string
    constraints: [ valid_values: [ MyCompanyVNFM ] ]
  default: [ MyCompanyVNFM ]
requirements:
- virtual_link_backend:
  capability: tosca.capabilities.nfv.VirtualLinkable
- virtual_link_service:
  capability: tosca.capabilities.nfv.VirtualLinkable
interfaces:
  Vnflcm:
    type: tosca.interfaces.nfv.Vnflcm
    instantiate:
      inputs:
        additional_parameters:
          type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters

```

The vnf node template in the sunshine.vnfd.tosca.yaml file is abstract and is subject to substitution; the lower-level templates in the subsequent sections provide these substitutions. The actual lower-level template is selected based on the node type and a value constraint on the flavour_id property.

Each low level service template contains a node template of type MyCompany.SunshineDB.1_0.1_0 with implementation of the LCM interfaces.

SunshineDB (simple): Lower level

sunshinedbsimple.vnfd.tosca.yaml

This example illustrates one Vdu.Compute nodes (dbBackend) with two connection points and two virtual links (see figure A.2-2). The flavour_id is "Simple".

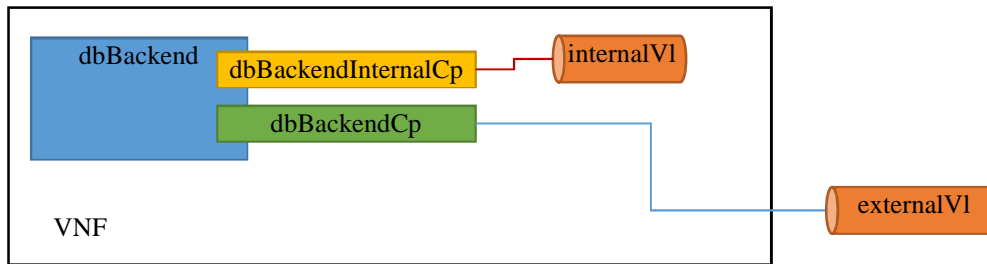


Figure A.2-2: SunshineDB (simple): Lower level

```

tosca_definitions_version: tosca_simple_yaml_1_2

description: Relational database, simple
imports:
  - etsi_nfv_sol001_vnfd_2_5_1_types.yaml # all of TOSCA types as defined in NFV SOL
001 GS
  - sunshineVNF.yaml # contains the VNF node type definition

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
  properties:
    ..
    flavour_id: simple
  requirements:
    virtual_link_backend: [ dbBackendCp, virtual_link ] # IPv4 for SQL

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    properties:
      flavour_description: A simple flavour
    interfaces:
      Vnflcm:
        instantiate:
          implementation: instantiate.workbook.mistral.yaml
        terminate:
          implementation: terminate.workbook.mistral.yaml

dbBackend:
  type: tosca.nodes.nfv.Vdu.Compute
  properties:
    name: dbBackend
    description: dbBackend compute node
    boot_order: []
    nfvi_constraints: []
    configurable_properties: {}
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
  capabilities:
    virtual_compute:
      properties:
        virtual_memory:
          virtual_mem_size: 8192 MiB
        virtual_cpu:
          cpu_architecture: x86
          num_virtual_cpu: 2
  
```

```

        virtual_cpu_clock: 1800 MHz
requirements:
  - virtual_storage: mariaDbStorage

mariaDbStorage:
  type: toasca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    virtual_block_storage_data:
      size_of_storage: 100 GB
      rdma_enabled: true
    sw_image_data:
      name: Software of Maria Db
      version: '1.0'
      checksum: 9af30fce37a4c5c831e095745744d6d2
      container_format: bare
      disk_format: qcow2
      min_disk: 2 GB
      min_ram: 8192 MiB
      size: 2 GB
      operating_system: Linux
      supported_virtualisation_environments:
        - KVM
    ..
  artifacts:
    sw_image:
      type: toasca.artifacts.nfv.SwImage
      file: maria.db.image.v1.0.qcow2

dbBackendCp:
  type: toasca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    role: leaf
    description: External connection point to access the DB on IPv4
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: false
  requirements:
    - virtual_binding: dbBackend
    # - virtual_link: # the target node is determined in the NSD

dbBackendInternalCp:
  type: toasca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    role: leaf
    description: Internal connection point on an VL
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: false
  requirements:
    - virtual_binding: dbBackend
    - virtual_link: internalVl

internalVl:
  type: toasca.nodes.nfv.VnfVirtualLink
  properties:
    connectivity_type:
      layer_protocols: [ ipv4 ]
      flow_pattern: mesh
    test_access: [ none ]
    description: Internal Virtual link in the VNF
    vl_profile:

```

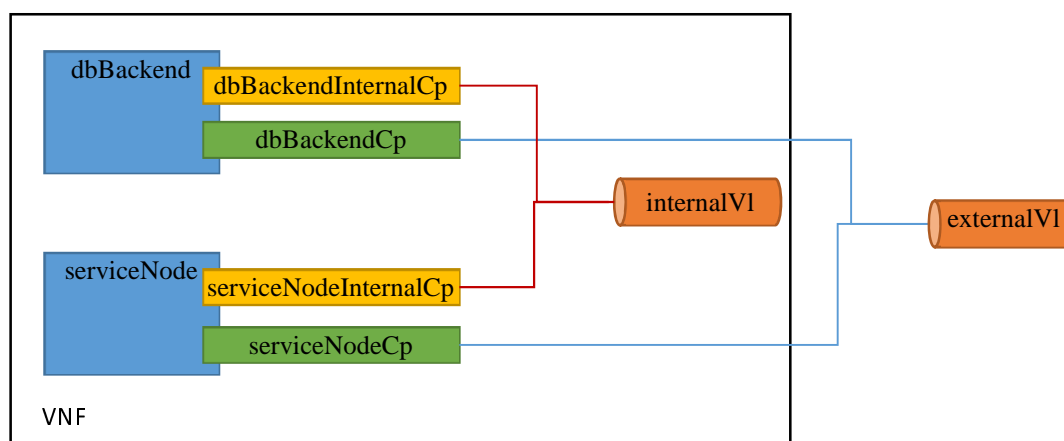
```

max_bitrate_requirements:
  root: 100000
  leaf: 20000
min_bitrate_requirements:
  root: 10000
  leaf: 10000

```

SunshineDB (complex): Lower level

This example illustrates two Vdu.Compute nodes (dbBackend, and serviceNode) with two connection points and two virtual links (see figure A.2-3). The flavour_id is "complex".



NOTE: The single external VL above illustrates that both serviceNodeCp and dbBackendCp are connected to the same external VL. Alternatively, external connection points can be connected to separate external VLs.

Figure A.2-3: SunshineDB (complex): lower level

sunshinedbcomplex.vnfd.tosca.yaml

```

tosca_definitions_version: tosca_simple_yaml_1_2

description: Relational database, complex
imports:
  - etsi_nfv_sol001_vnfd_2_5_1_types.yaml # all of TOSCA types as defined in NFV SOL
001 GS
  - sunshineVNF.yaml # contains the VNF node type definition

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
  properties:
    ..
    flavour_id: complex
  requirements:
    virtual_link_backend: [ dbBackendCp, virtual_link ] # IPv4 for SQL
    virtual_link_service: [ serviceNodeCp, virtual_link ] # IPv4 for SSH

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    properties:
      flavour_description: A complex flavour
    interfaces:

```



```
Vnflcm:
  instantiate:
    implementation: instantiate.workbook.mistral.yaml
  terminate:
    implementation: terminate.workbook.mistral.yaml
  heal:
    implementation: heal.workbook.mistral.yaml

dbBackend:
  type: toasca.nodes.nfv.Vdu.Compute
  properties:
    name: dbBackend
    description: dbBackend compute node
    boot_order: []
    nfvi_constraints: []
    configurable_properties: {}
    vdu_profile:
      min_number_of_instances: 3
      max_number_of_instances: 4
  capabilities:
    virtual_compute:
      properties:
        virtual_memory:
          virtual_mem_size: 8192 MiB
        virtual_cpu:
          cpu_architecture: x86
          num_virtual_cpu: 2
          virtual_cpu_clock: 1800 MHz
  requirements:
    - virtual_storage: mariaDbStorage

serviceNode:
  type: toasca.nodes.nfv.Vdu.Compute
  properties:
    name: serviceNode
    description: brief description about serviceNode
    boot_order: []
    nfvi_constraints: []
    configurable_properties: {}
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
  sw_image_data:
    name: Software of Maria Db
    version: '1.0'
    checksum: 9af30fce37a4c5c831e095745744d6d2
    container_format: bare
    disk_format: qcow2
    min_disk: 2 GB
    min_ram: 8192 MiB
    size: 2 GB
    operating_system: Linux
    supported_virtualisation_environments:
      - KVM
  capabilities:
    virtual_compute:
      properties:
        virtual_memory:
          virtual_mem_size: 8192 MiB
        virtual_cpu:
          cpu_architecture: x86
```

```
        num_virtual_cpu: 2
        virtual_cpu_clock: 1800 MHz
requirements:
  - virtual_storage: mariaDbStorage
artifacts:
  sw_image:
    type: tosca.artifacts.nfv.SwImage
    file: maria.db.image.v1.0.qcow2

mariaDbStorage:
  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    virtual_block_storage_data:
      size_of_storage: 100 GB
      rdma_enabled: true
    sw_image_data:
      name: Software of Maria Db
      version: '1.0'
      checksum: 9af30fce37a4c5c831e095745744d6d2
      container_format: bare
      disk_format: qcow2
      min_disk: 2 GB
      min_ram: 8192 MiB
      size: 2 GB
      operating_system: Linux
      supported_virtualisation_environments:
        - KVM
  artifacts:
    sw_image:
      type: tosca.artifacts.nfv.SwImage
      file: maria.db.image.v1.0.qcow2

dbBackendCp:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    role: leaf
    description: External connection point to access the DB on IPv4
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: false
  requirements:
    - virtual_binding: dbBackend
    #- virtual_link: # the target node is determined in the NSD

dbBackendInternalCp:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    role: leaf
    description: Internal connection point on an VL
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: false
  requirements:
    - virtual_binding: dbBackend
    - virtual_link: internalVL

serviceNodeCp:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    role: leaf
```

```

description: External connection point to access the DB on IPv4
protocol: [ associated_layer_protocol: ipv4 ]
trunk_mode: false
requirements:
- virtual_binding: serviceNode
  #- virtual_link: # the target node is determined in the NSD

serviceNameInternalCp:
type: toasca.nodes.nfv.VduCp
properties:
  layer_protocols: [ ipv4 ]
  role: leaf
  description: Internal connection point on VL
  protocol: [ associated_layer_protocol: ipv4 ]
  trunk_mode: false
requirements:
- virtual_binding: serviceNode
- virtual_link: internalVl

internalVl:
type: toasca.nodes.nfv.VnfVirtualLink
properties:
  connectivity_type:
    layer_protocols: [ ipv4 ]
    flow_pattern: mesh
  test_access: [ none ]
  description: Internal VL
  vl_profile:
    max_bitrate_requirements:
      root: 100000
      leaf: 20000
    min_bitrate_requirements:
      root: 10000
      leaf: 10000

```

A.3 VNF external connection point

A.3.1 General

A VNF external connection point may either be an internal connection point (a VDU connection point) which is re-exposed externally, i.e. it may be connected to an external virtual link, or a virtual link port. In the latter case the external connection point node has an association relationship of type `VirtualLinksTo` to the internal virtual link node.

The following clauses illustrate the use of both models of VNF external connection point, re-exposure of an internal connection point and connected to an internal virtual link.

A.3.2 External connection point re-exposing an internal connection point

In this case there is no need for a `VnfExtCp` node template. When substituting the VNFD low level service template for the abstract VNF node, the `virtual_link` requirement of the abstract VNF node is mapped to the VDU connection point's `virtual_link` requirement. This is shown in figure A.3.2-1. In this case the VNF Service Template does not include a node template of type `tosca.nodes.nfv.VnfExtCp`, but the functionality of it is provided by a VDU connection point.

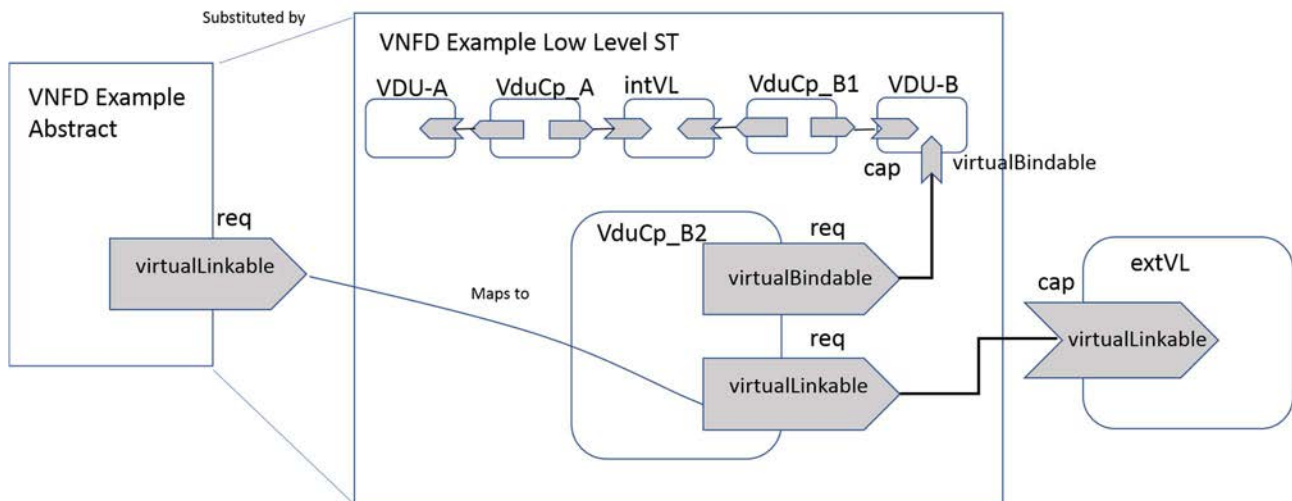


Figure A.3.2-1: VNFD with an VDU connection point acting as VnfExtCp

The following snippet shows the relevant part of the service template:

```

tosca_definitions_version: tosca_simple_yaml_1_2

node_types:
  tosca.nodes.nfv.exampleVNF:
    derived from: tosca.nodes.nfv.VNF
    properties:
      # omitted for brevity
    requirements:
      - virtual_link:
          capability: tosca.capabilities.nfv.VirtualLinkable
    ...

topology_template:
  ...

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    requirements:
      virtual_link: [vduCp_B2, virtual_link]

  node_templates:
    vduCp_B2:
      type: tosca.nodes.nfv.VduCp
      # properties omitted for brevity
      requirements:
        - virtual_binding: VDU-B
        - virtual_link: # mapped to virtual_link requirement of VNF node

    # other node template definitions (VDU-A, VDU-B, intVL, etc.):
    ...

```

A.3.3 External connection point connected to an internal virtual link

In this case a VnfExtCp node template is needed. When substituting the VNFD low level service template for the abstract VNF node type, the virtual_link requirement of the abstract VNF node maps to the external_virtual_link requirement of the VnfExtCp node. The internal_virtual_link requirement of the VnfExtCp node is fulfilled with the corresponding capability of the internal VirtualLink node. This is shown in figure A.3.3-1.

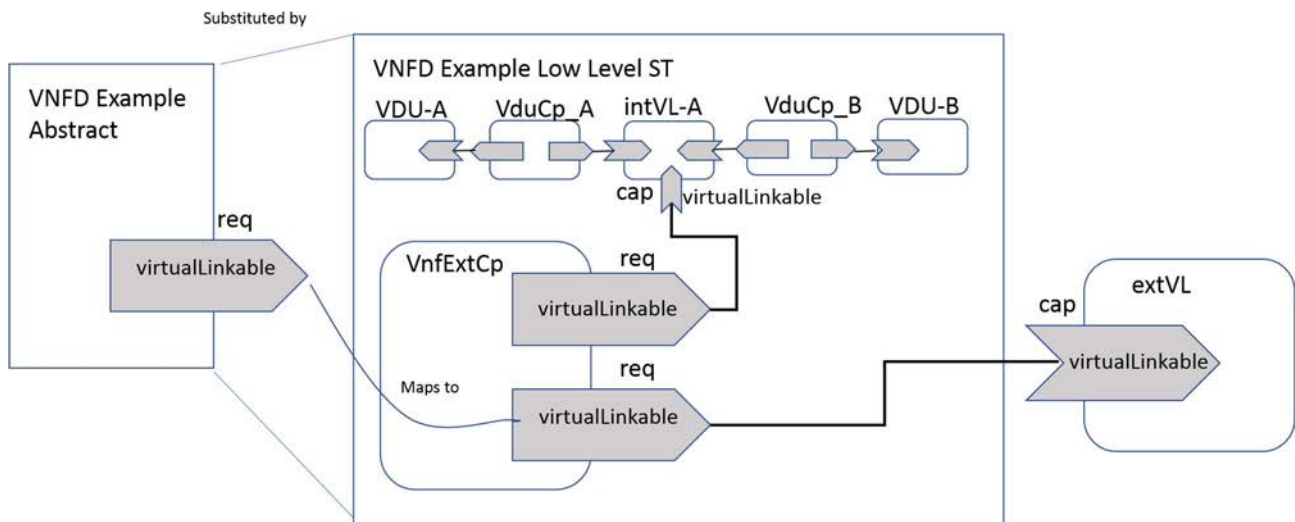


Figure A.3.3-1: VNFD with a VnfExtCp connected to an internal virtual link

The following snippet shows the corresponding node template definition:

```

tosca_definitions_version: tosca_simple_yaml_1_2
node_types:
  tosca.nodes.nfv.exampleVNF:
    derived from: tosca.nodes.nfv.VNF
    properties:
      # omitted for brevity
    requirements:
      - virtual_link:
          capability: tosca.capabilities.nfv.VirtualLinkable

topology_template:
  ...

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    requirements:
      virtual_link: [myMRFExtCp, external_virtual_link]

  node_templates:
    myMRFExtCp:
      type: tosca.nodes.nfv.VnfExtCp
      # properties omitted for brevity
      ...
      requirements:
        - external_virtual_link: # mapped to virtual_link requirement of VNF node
        - internal_virtual_link: intVL-A

    intVL-A:
      type: tosca.nodes.nfv.VnfVirtualLink
      properties:

```

```

connectivity_type:
  layer_protocols: [ ipv4 ]
  flow_pattern: mesh
description: Internal VL
vl_profile:
  max_bitrate_requirements:
    root: 1000000
    leaf: 100000
  min_bitrate_requirements:
    root: 100000
    leaf: 10000
  qos:
    latency: 100 ms
    packet_delay_variation: 80 ms
    packet_loss_ratio: 0.00001
capabilities:
  virtual_linkable:
  monitoring_parameter:

# Other node templates e.g. VDU-A, VDU-B, VduCP_A, etc.:
...

```

A.4 VNFD modeling design example by using TOSCA composition

The following example in figure A.4-1 shows a VNFD descriptor contains three VDUs, which are interconnected by two virtualLinks.

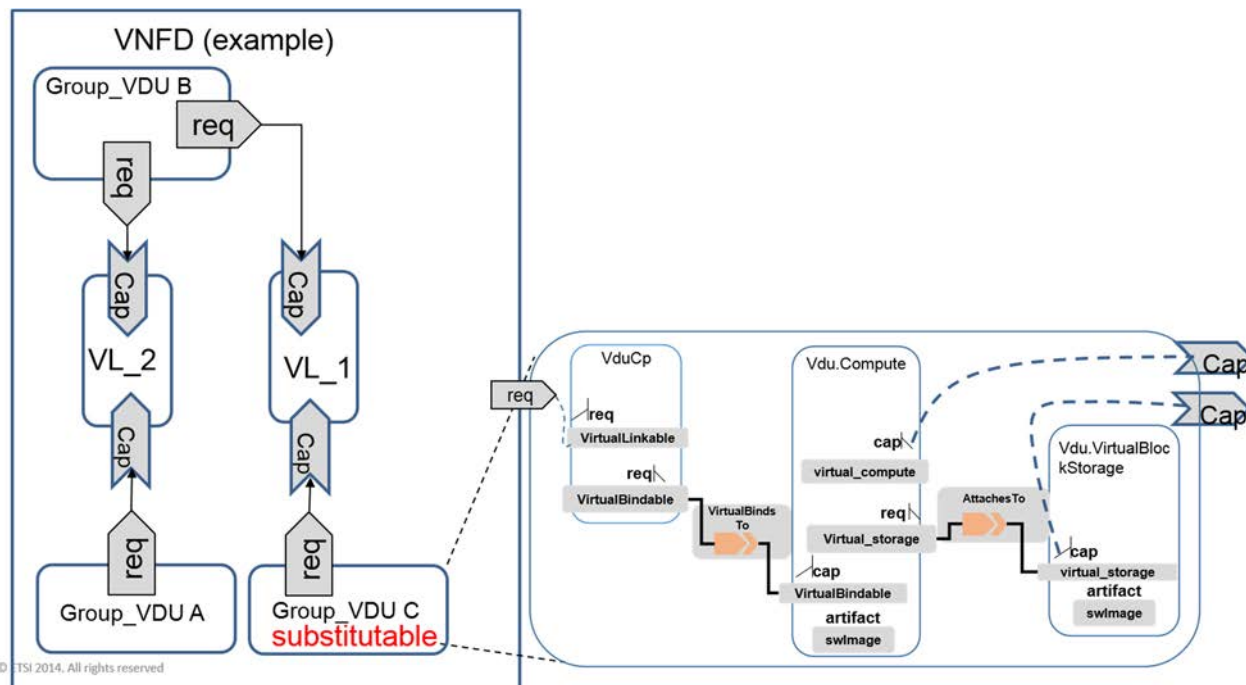


Figure A.4-1: Example of using substitution mapping for the VNFD design

In this example, a separate service template is used to describe the composition of Vdu.Compute, Vdu.VirtualBlockStorage and VduCp, and then substituted as a node template in a VNFD service template. `tosca.nodes.nfv.groupVDU_A`, `tosca.nodes.nfv.groupVDU_B`, `tosca.nodes.nfv.groupVDU_C` types are used for substitution_mapping.

```
tosca_definitions_version: tosca_simple_yaml_1_2

topology_template:
  description: Template of a VNFD example

  node_templates:
    Group_VDU_A:
      type: tosca.nodes.nfv.groupVDU_A
      requirements:
        - virtual_link:VL_2

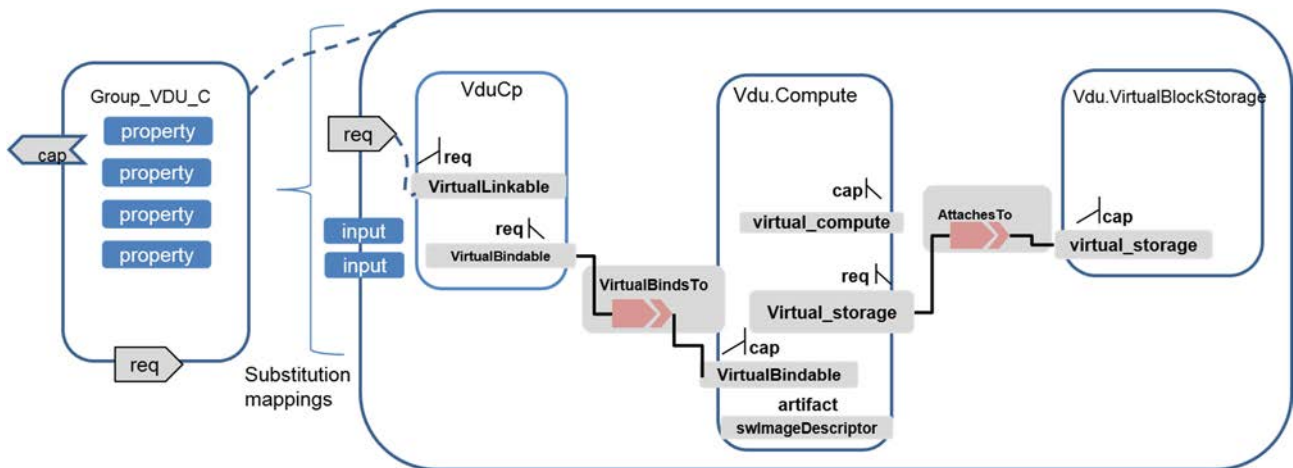
    Group_VDU_B:
      type: tosca.nodes.nfv.groupVDU_B
      requirements:
        - virtual_link: VL_2
        - virtual_link1: VL_1

    Group_VDU_C:
      type: tosca.nodes.nfv.groupVDU_C # the description of this type is described
                                      # in another service template.
      requirements:
        - virtual_link:VL_1

    VL_1:
      type: tosca.nodes.nfv.VnfVirtualLink
      properties:
        # omitted here for brevity
      capabilities:
        virtual_linkable:

    VL_2:
      type: tosca.nodes.nfv.VnfVirtualLink
      properties:
        # omitted here for brevity
      capabilities:
        virtual_linkable:
```

The TOSCA service template example of Group_VDU C is showing in figure A.4-2.



© ETSI 2014. All rights reserved

Figure A.4-2: Example of composition Vdu.Compute, Vdu.VirtualBlockStorage and VduCp together using TOSCA substitution mapping

```

tosca_definitions_version: tosca_simple_yaml_1_2

description: composition of Vdu.Compute, Vdu.VirtualBlockStorage and VduCp.

topology_template:
  substitution_mappings:
    node_type: tosca.nodes.nfv.groupVDU_C # substituted as a node type
  requirements:
    virtual_link: [ internalCpd, virtual_link ]
  capabilities:
    virtual_compute: [ vduC_compute, virtual_compute ]
    virtual_storage: [ vduC_storage, virtual_storage ]

  node_templates:
    vduC_compute:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        # omitted here for brevity
      capabilities:
        virtual_compute:
      artifacts:
        sw_image:
      requirements:
        - virtual_storage: vduC_storage

    vduC_storage:
      type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
      properties:
        # omitted here for brevity
      capabilities:
        virtual_storage:

    internalCpd:
      type: tosca.nodes.nfv.VduCp
      properties:
        # omitted here for brevity
      requirements:
        - virtual_linkable:
        - virtual_binding: vduC_compute
  
```


A.5 VNFD with Single deployment flavour modelling design example

In this example, there is one deployment flavour applied to this VNFD, and TOSCA-Simple-Profile-yaml-v1.1 is used for designing and processing this VNFD TOSCA model.

The one service template design illustrated by this example is only applicable when the VNF has only one deployment flavour.

The service template is the main entry point in the VNF Package, i.e. the Entry-definitions file, and is deployed as a stand-alone service template, i.e. without substituting for a node template. However, the service template still contains substitution_mappings to indicate its ability to substitute for a node template of the specific node type.

The service template contains a node template of type MyCompany.SunshineDB.1_0.1_0. The node template contains the properties defined in the node type definition and implementations for the LCM interfaces.

sunshinesimple.vnfd.tosca.yaml

This example illustrates a VNF with one VDU.Compute nodes (dbBackend) with two VDU connection points and one VNF virtual link (see figure A.5-1). The VNF is also connected to an external virtual link. The flavour_id is "simple".

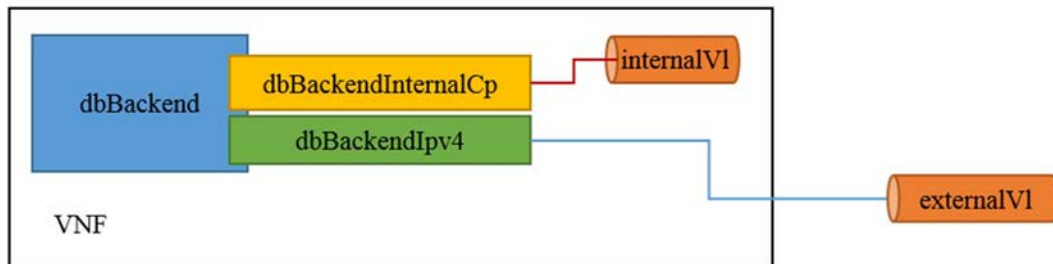


Figure A.5-1: SunshineDB (simple)

```
tosca_definitions_version: tosca_simple_yaml_1_1

description: Relational database, simple

imports:
  - etsi_nfv_sol001_vnfd_2_5_1_types.yaml # all of TOSCA types as defined in NFV SOL
001 GS for a VNFD

data_types:
  MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      parameter_1:
        type: string
        required: true
        default: value_1
      parameter_2:
        type: string
        required: false
        default: value_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
```

```

    type: string
    constraints: [ valid_values: [ b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ] ]
    default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
  provider:
    type: string
    constraints: [ valid_values: [ MyCompany ] ]
    default: MyCompany
  product_name:
    type: string
    constraints: [ valid_values: [ SunshineDB ] ]
    default: SunshineDB
  software_version:
    type: string
    constraints: [ valid_values: [ '1.0' ] ]
    default: '1.0'
  descriptor_version:
    type: string
    constraints: [ valid_values: [ '1.0' ] ]
    default: '1.0'
  flavour_id:
    type: string
    constraints: [ valid_values: [ simple ] ]
    default: simple
  flavour_description:
    type: string
    default: ""
  vnfm_info:
    type: list
    entry_schema:
      type: string
      constraints: [ valid_values: [ [ MyCompanyVNFM ] ] ]
    default: [MyCompanyVNFM ]
  interfaces:
    Vnflcm:
      type: toasca.interfaces.nfv.Vnflcm
      instantiate:
        inputs:
          additional_parameters:
            type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters
      #terminate:

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
  requirements:
    virtual_link: [ dbBackendIpv4, virtual_link ] # IPv4 for SQL

node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    properties:
      descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      provider: Mycompany
      product_name: SunshineDB
      software_version: '1.0'
      descriptor_version: '1.0'
      vnfm_info:
        - Mycompany
      flavour_id: simple
      flavour_description: 'vnf simple flavour description'
    interfaces:

```

```

Vnflcm:
  instantiate:
    implementation: instantiate.workbook.mistral.yaml
  terminate:
    implementation: terminate.workbook.mistral.yaml
  heal:
    implementation: heal.workbook.mistral.yaml

dbBackend:
  type: toasca.nodes.nfv.Vdu.Compute
  properties:
    name: dbbackend
    description: dbBackend
    boot_order: []
    nfvi_constraints: []
    configurable_properties: {}
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 4
  capabilities:
    virtual_compute:
      properties:
        virtual_memory:
          virtual_mem_size: 8192 MiB
        virtual_cpu:
          cpu_architecture: x86
          num_virtual_cpu: 2
          virtual_cpu_clock: 1800 MHz
  requirements:
    - virtual_storage: mariaDbStorage

mariaDbStorage:
  type: toasca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    virtual_block_storage_data:
      size_of_storage: '200 GB'
      rdma_enabled: true
    sw_image_data:
      name: Software of Maria Db
      version: '1.0'
      checksum: 9af30f3e37a4c5c831e095745744d6d2
      container_format: bare
      disk_format: qcow2
      min_disk: 2 GB
      min_ram: 8192 MiB
      size: 2 GB
      operating_system: Linux
      supported_virtualisation_environments:
        - KVM
  artifacts:
    sw_image:
      type: toasca.artifacts.nfv.SwImage
      file: maria.db.image.v1.0.qcow2

dbBackendInternalCp:
  type: toasca.nodes.nfv.VduCp
  properties:
    protocol: [associated_layer_protocol: ipv4 ]
    trunk_mode: false
    layer_protocols: [ ipv4 ]
    role: leaf

```

```

description: Internal connection point on an VL
requirements:
- virtual_binding: dbBackend
- virtual_link: internalVl

internalVl:
type: toasca.nodes.nfv.VnfVirtualLink
properties:
connectivity_type:
layer_protocols: [ ipv4 ]
flow_pattern: mesh
test_access: [ ]
description: Internal Virtual link in the VNF
vl_profile:
max_bitrate_requirements:
root: 100000
leaf: 20000
min_bitrate_requirements:
root: 10000
leaf: 10000

dbBackendIpv4:
type: toasca.nodes.nfv.VduCp
properties:
protocol: [ associated_layer_protocol: ipv4 ]
trunk_mode: false
layer_protocols: [ ipv4 ]
role: leaf
description: External connection point to access the DB on IPv4
requirements:
#- virtual_link: # the target node is determined in the NSD

- virtual_binding: dbBackend

```

A.6 Scaling and Instantiation Level examples

A.6.1 ScalingAspect and InstantiationLevels policies with uniform delta

This example shows instantiationLevels, and ScalingAspect policies types in the complex scaling scenario where scaling aspect delta is based on "uniform delta" values.

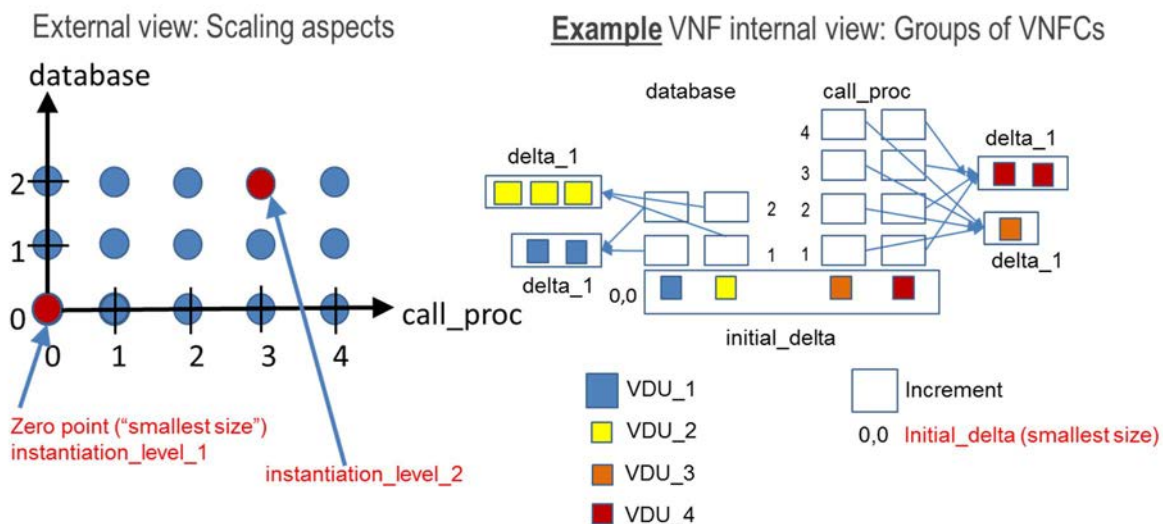


Figure A.6.1-1: Complex scaling example with uniform delta

```
tosca_definitions_version: tosca_simple_yaml_1_2
```

```
description: Complex scaling example (uniform delta value) described with policies
```

```
topology_template:
```

```
  node_templates:
```

```
    vdu_1:
```

```
      type: tosca.nodes.nfv.VDU.Compute
```

```
      properties:
```

```
        name: ..
```

```
        description: ..
```

```
        boot_order:
```

```
          - ..
```

```
        vdu_profile:
```

```
          min_number_of_instances: 1
```

```
          max_number_of_instances: 5
```

```
        nfvi_constraints:
```

```
          - ..
```

```
        configurable_properties:
```

```
          additional_vnfc_configurable_properties: {}
```

```
    vdu_2:
```

```
      type: tosca.nodes.nfv.VDU.Compute
```

```
      properties:
```

```
        name: ..
```

```
        description: ..
```

```
        boot_order:
```

```
          - ..
```

```
        vdu_profile:
```

```
          min_number_of_instances: 1
```

```
          max_number_of_instances: 7
```

```
        nfvi_constraints:
```

```
          - ..
```

```
        configurable_properties:
```

```
          additional_vnfc_configurable_properties: {}
```

```
    vdu_3:
```

```
      type: tosca.nodes.nfv.VDU.Compute
```

```
      properties:
```

```
        name: ..
```

```

description: ..
boot_order:
  - ..
vdu_profile:
  min_number_of_instances: 1
  max_number_of_instances: 5
nfvi_constraints:
  - ..
configurable_properties:
  additional_vnfc_configurable_properties: {}

vdu_4:
type: toasca.nodes.nfv.VDU.Compute
properties:
  name: ..
  description: ..
  boot_order:
    - ..
  vdu_profile:
    min_number_of_instances: 1
    max_number_of_instances: 9
  nfvi_constraints:
    - ..
  configurable_properties:
    additional_vnfc_configurable_properties: {}

vl_1:
type: toasca.nodes.nfv.VnfVirtualLink
properties:
  connectivity_type:
    layer_protocol: ipv4
  vl_profile:
    min_bitrate_requirements:
      root: 1000000
    max_bitrate_requirements:
      root: 2000000

vl_2:
type: toasca.nodes.nfv.VnfVirtualLink
properties:
  connectivity_type:
    layer_protocol: ipv4
  vl_profile:
    min_bitrate_requirements:
      root: 1000000
    max_bitrate_requirements:
      root: 4000000

policies:
- scaling_aspects:
  type: toasca.policies.nfv.ScalingAspects
  properties:
    aspects:
      database:
        name: ..
        description: ..
        max_scale_level: 2
        step_deltas:
          - delta_1
    call_proc:
      name: ..

```

```
        description: ..
        max_scale_level: 4
        step_deltas:
          - delta_1

- vdu_1_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 1
  targets: [ vdu_1 ]

- vdu_1_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: database
    deltas:
      delta_1:
        number_of_instances: 2
  targets: [ vdu_1 ]

- vdu_2_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 1
  targets: [ vdu_2 ]

- vdu_2_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingDeltas
  properties:
    aspect: database
    deltas:
      delta_1:
        number_of_instances: 3
  targets: [ vdu_2 ]

- vl_1_bitrate_initial_delta:
  type: tosca.policies.nfv.VirtualLinkBitrateInitialDelta
  properties:
    initial_delta:
      bitrate_requirements:
        root: 1000000
  targets: [ vl_1 ]

- vl_1_bitrate_scaling_aspect_deltas:
  type: tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas
  properties:
    aspect: database
    deltas:
      delta_1:
        bitrate_requirements:
          root: 1000000
  targets: [ vl_1 ]

- vdu_3_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
```

```

    initial_delta:
      number_of_instances: 1
    targets: [ vdu_3 ]

- vdu_3_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: call_proc
    deltas:
      delta_1:
        number_of_instances: 1
    targets: [ vdu_3 ]

- vdu_4_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 1
    targets: [ vdu_4 ]

- vdu_4_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: call_proc
    deltas:
      delta_1:
        number_of_instances: 2
    targets: [ vdu_4 ]

- vl_2_bitrate_initial_delta:
  type: tosca.policies.nfv.VirtualLinkBitrateInitialDelta
  properties:
    initial_delta:
      bitrate_requirements:
        root: 1000000
    targets: [ vl_2 ]

- vl_2_bitrate_scaling_aspect_deltas:
  type: tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas
  properties:
    aspect: call_proc
    deltas:
      delta_1:
        bitrate_requirements:
          root: 1000000
    targets: [ vl_2 ]

- instantiation_levels:
  type: tosca.policies.nfv.InstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        description: ..
        scale_info:
          database:
            scale_level: 0
          call_proc:
            scale_level: 0
      instantiation_level_2:
        description: ..
        scale_info:

```



```
        database:
          scale_level: 2
        call_proc:
          scale_level: 3
      default_level: instantiation_level_1

- vdu_1_instantiation_levels:
  type: tosca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
      instantiation_level_2:
        number_of_instances: 5
  targets: [ vdu_1 ]

- vdu_2_instantiation_levels:
  type: tosca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
      instantiation_level_2:
        number_of_instances: 7
  targets: [ vdu_2 ]

- vl_1_instantiation_levels:
  type: tosca.policies.nfv.VirtualLinkInstantiationLevels
  ..
  targets: [ vl_1 ]

- vdu_3_instantiation_levels:
  type: tosca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
      instantiation_level_2:
        number_of_instances: 4
  targets: [ vdu_3 ]

- vdu_4_instantiation_levels:
  type: tosca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
      instantiation_level_2:
        number_of_instances: 7
  targets: [ vdu_4 ]

- vl_2_instantiation_levels:
  type: tosca.policies.nfv.VirtualLinkInstantiationLevels
  ..
  targets: [ vl_2 ]
```

A.6.2 ScalingAspect and InstantiationLevelspolicies with non-uniform deltas

This example shows instantiationLevel, ScalingAspect policies and group types in the complex scaling scenario where scaling aspect delta is based on "delta" (non-uniform) values for Processing Auxiliary VNFC.

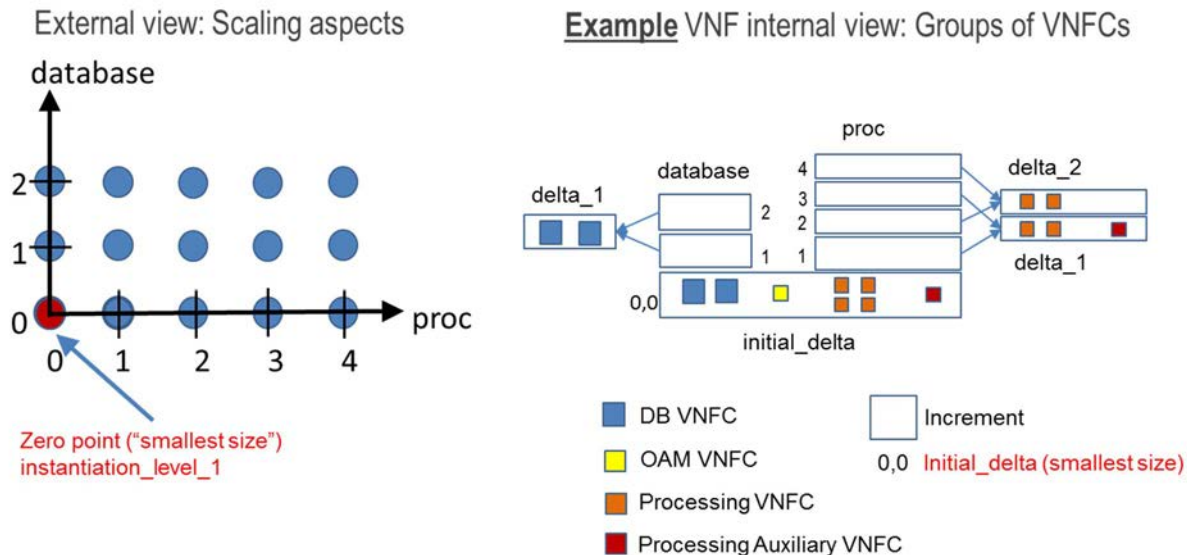


Figure A.6.2-1: Complex scaling example with non-uniform delta

For simplicity, virtual link is not illustrated and instantiation level is one level (default) in this example, as these properties are already illustrated in clause A.6.1.

The "Proc" aspect contains a uniform delta part ("processing" and a non-uniform part ("processing_auxiliary").

The "DB" aspect contains a non-uniform part ("db"), and no delta information for "oam".

```
tosca_definitions_version: tosca_simple_yaml_1_2

description: Complex example (uniform and non-uniform delta value) described with
policies

topology_template:
  node_templates:
    db:
      type: tosca.nodes.nfv.VDU.Compute
      properties:
        name: ..
        description: ..
        boot_order:
          - ..
        vdu_profile:
          min_number_of_instances: 2
          max_number_of_instances: 6
        nfvi_constraints:
          - ..
        configurable_properties:
          additional_vnfc_configurable_properties: {}

    oam:
      type: tosca.nodes.nfv.VDU.Compute
      properties:
        name: ..
```

```

description: ..
boot_order:
  - ..
vdu_profile:
  min_number_of_instances: 1
  max_number_of_instances: 1
nfvi_constraints:
  - ..
configurable_properties:
  additional_vnfc_configurable_properties: {}

processing:
type: toasca.nodes.nfv.VDU.Compute
properties:
  name: ..
  description: ..
  boot_order:
    - ..
  vdu_profile:
    min_number_of_instances: 4
    max_number_of_instances: 12
  nfvi_constraints:
    - ..
  configurable_properties:
    additional_vnfc_configurable_properties: {}

processing_auxiliary:
type: toasca.nodes.nfv.VDU.Compute
properties:
  name: ..
  description: ..
  boot_order:
    - ..
  vdu_profile:
    min_number_of_instances: 1
    max_number_of_instances: 3
  nfvi_constraints:
    - ..
  configurable_properties:
    additional_vnfc_configurable_properties: {}

policies:
- scaling_aspects:
type: toasca.policies.nfv.ScalingAspects
properties:
  aspects:
    database:
      name: ..
      description: ..
      max_scale_level: 2
      step_deltas:
        - delta_1
        - delta_1
    proc:
      name: ..
      description: ..
      max_scale_level: 4
      step_deltas:
        - delta_1
        - delta_2
        - delta_1

```

```
- delta_2

- db_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 2
  targets: [ db ]

- db_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: database
    deltas:
      delta_1:
        number_of_instances: 2
  targets: [ db ]

- oam_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 1
  targets: [ oam ]

- processing_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 4
  targets: [ processing ]

- processing_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: proc
    deltas:
      delta_1:
        number_of_instances: 2
      delta_2:
        number_of_instances: 2
  targets: [ processing ]

- processing_auxiliary_initial_delta:
  type: tosca.policies.nfv.VduInitialDelta
  properties:
    initial_delta:
      number_of_instances: 1
  targets: [ processing_auxiliary ]

- processing_auxiliary_scaling_aspect_deltas:
  type: tosca.policies.nfv.VduScalingAspectDeltas
  properties:
    aspect: proc
    deltas:
      delta_1:
        number_of_instances: 1
      delta_2:
        number_of_instances: 0
  targets: [ processing_auxiliary ]
```

```

- instantiation_levels:
  type: toasca.policies.nfv.InstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        description: ..
        scale_info:
          database:
            scale_info:
              scale_level: 0
          proc:
            scale_info:
              scale_level: 0

- db_instantiation_levels:
  type: toasca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 2
  targets: [ db ]

- oam_instantiation_levels:
  type: toasca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
  targets: [ oam ]

- processing_instantiation_levels:
  type: toasca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 4
  targets: [ processing ]

- processing_auxiliary_instantiation_levels:
  type: toasca.policies.nfv.VduInstantiationLevels
  properties:
    levels:
      instantiation_level_1:
        number_of_instances: 1
  targets: [ processing_auxiliary ]

```

A.7 Service Access Point

A.7.1 General

A SAP may either be a VNF external connection point (or a PNF external connection points or a Sap of a nested NS of this NS) which is exposed as CP of one of the NS constituents (VnfExtCp, PnfExtCp, nested NS Sap externally, or be exposed as a new CP on NS virtual link. This is shown in figure A.7.1-1. In the latter case the Sap node has an association relationship of type VirtualLinksTo to the NsVirtualLink node.

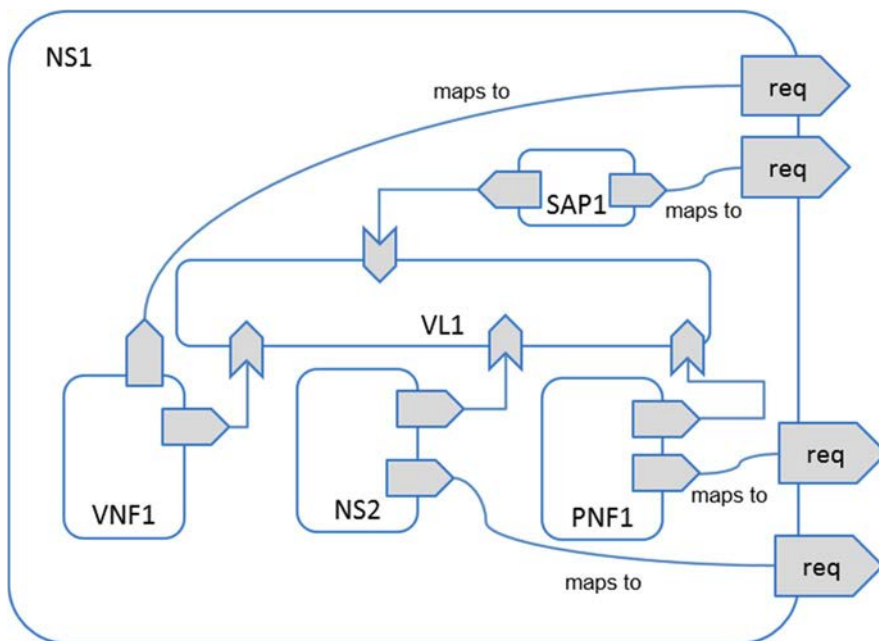


Figure A.7.1-1: Overview of SAP options

The following clauses illustrate the use of both models of SAP, re-exposure of a VNF external connection point and connected to an NS virtual link.

A.7.2 VNF External connection point exposing as a SAP

In this case, Sap node template is not required. When substituting the NSD service template for the abstract NS node, the virtual_link requirement of the abstract NS node is mapped to the VNF external connection point's virtual_link requirement. This is shown in figure A.7.2-1. In this case the NS Service Template does not include a node template of type `tosca.nodes.nfv.Sap`, but the functionality of it is provided by a VNF external connection point.

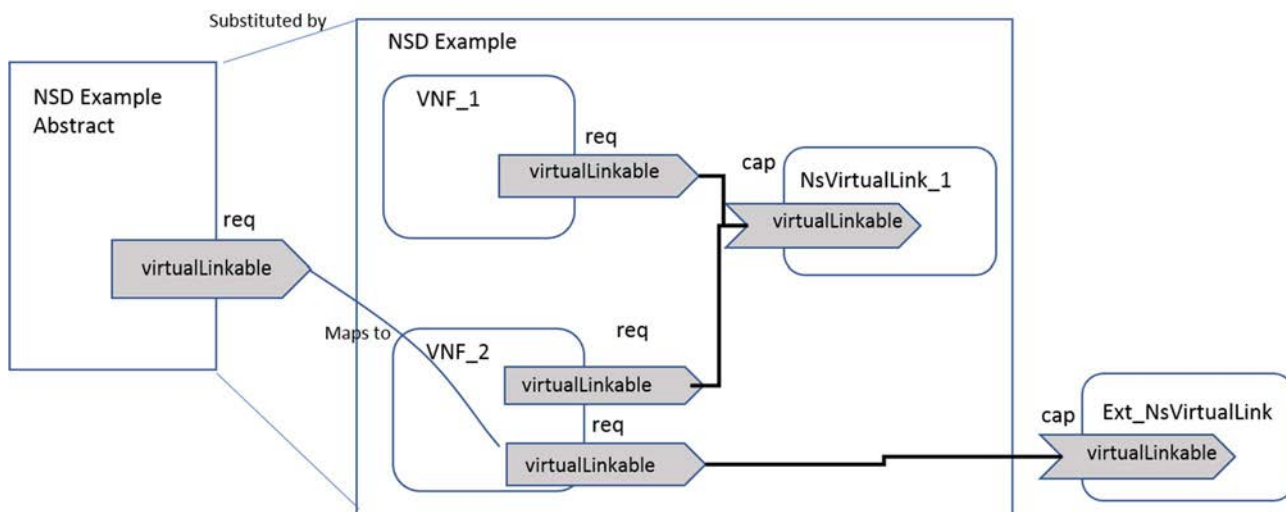


Figure A.7.2-1: VNF_2 External connection point (VnfExtCp) exposed as a Sap

The following snippet shows the relevant part of the NS service template:

```
tosca_definitions_version: toscasimpleyaml_1_2

node_types:
  toscanodes.nfv.exampleNS
```

```

derived from: toasca.nodes.nfv.NS
properties:
# omitted for brevity
requirements:
- virtual_link:
  capability:
    toasca.capabilitites.nfv.VirtualLinkable
...
topology_template:
...
substitution_mappings:
node_type: toasca.nodes.nfv.exampleNS
requirements:
  virtual_link: [VNF_2, virtual_link_2] # the External connection point of VNF_2
                                         # will be used as the Sap of this NS
node_templates:
VNF_2:
type: toasca.nodes.nfv.exampleVNF2
# properties omitted for brevity
requirements:
- virtual_link_1: NsVirtualLink_1 # connects to the External connection
                                  # point which maps to the
                                  # virtual_link_1 requirement of VNF_2
- virtual_link_2: # mapped to virtual_link requirement of NS node

# other node template definitions (VNF_1, NsVirtualLink_1 etc.):
...

```

A.7.3 SAP connected to an NS virtual link

In this case, a Sap node template is required. When substituting the NSD service template for the abstract NS node type, the `virtual_link` requirement of the abstract NS node maps to the `virtual_link` requirement of the Sap node. This is shown in figure A.7.3-1.

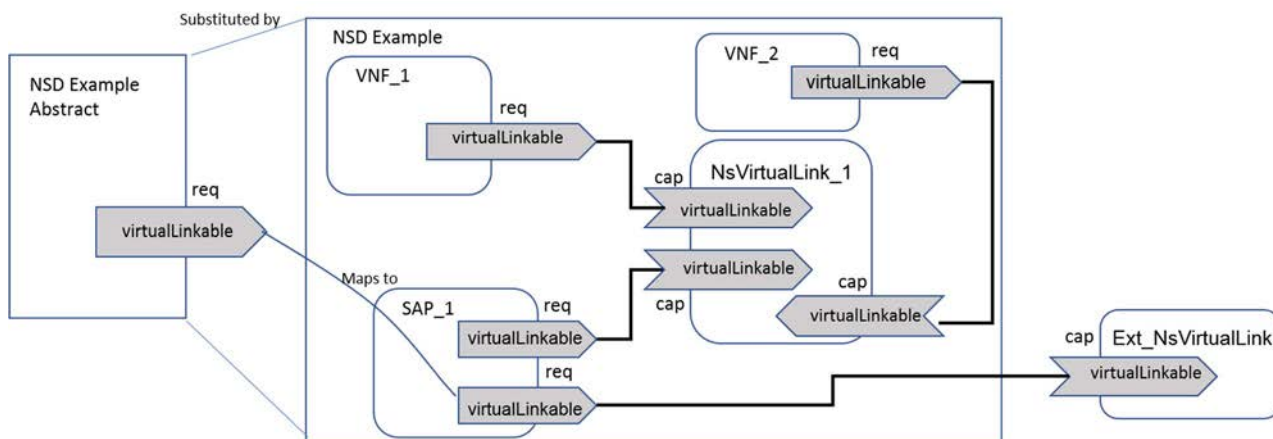


Figure A.7.3-1: SAP_1 exposed as new CP on an internal virtual link

The following snippet shows the corresponding node template definition:

```
tosca_definitions_version: toska_simple_yaml_1_2
node_types:
  toska.nodes.nfv.exampleNS
    derived from: toska.nodes.nfv.NS
    properties:
      # omitted for brevity
    requirements:
      - virtual_link:
          capability:
            toska.capabilitites.nfv.VirtualLinkable

topology_template:
...

substitution_mappings:
  node_type: toska.nodes.nfv.exampleNS
  requirements:
    virtual_link: [SAP_1, external_virtual_link_2] #SAP_1 is the Sap of the NS

node_templates:
  SAP_1:
    type: toska.nodes.nfv.Sap
    # properties omitted for brevity
    ...
    requirements:
      internal_virtual_link_1: NsVirtualLink_1
      external_virtual_link_2: # map to virtual_link requirement of the NS node

# Other node templates e.g. VNF_1, VNF_2, NsVirtualLink_1, etc.:
...
```

A.8 NSD with Single deployment flavour modelling design example

In this example, there is one deployment flavour applied to this NSD, and TOSCA-Simple-Profile-yaml-v1.1 is used for designing and processing this NSD TOSCA model.

The one service template design illustrated by this example is only applicable when the NS has only one deployment flavour.

The service template is the main entry point in the NSD file structure, i.e. the Entry-definitions file, and is deployed as a stand-alone service template, i.e. without substituting for a node template. However, the service template still contains substitution_mappings to indicate its ability to substitute for a node template of the specific node type.

example_NS.yaml

This example illustrates a NS contains two VNF: VNF_1 and VNF_2, they connect through NsVirtualLink_1. One of the VnfExtCp of VNF_2 is exposed as the Sap of this NS. The flavour_id is "simple".

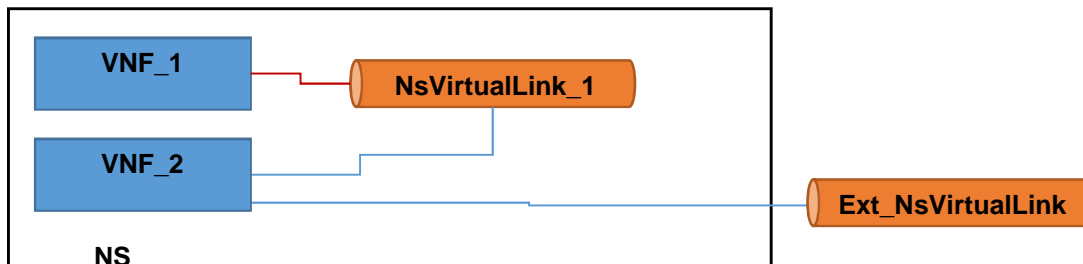


Figure A.8-1: example_NS (simple)

```

tosca_definitions_version: tosca_simple_yaml_1_1

description: Relational database, simple

imports:
  - etsi_nfv_sol001_nsd_2_5_1_types.yaml # all of NSD related TOSCA types as defined
in NFV SOL 001 GS

node_types:
  tosca.example_NS:
    derived_from: tosca.nodes.nfv.NS
    properties:
      descriptor_id:
        type: string
        constraints: [ valid_values: [ b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ] ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      designer:
        type: string
        constraints: [ valid_values: [ MyCompany ] ]
        default: MyCompany
      name:
        type: string
        constraints: [ valid_values: [ ExampleService ] ]
        default: ExampleService
      version:
        type: string
        constraints: [ valid_values: [ '1.0' ] ]
        default: '1.0'
      invariant_id:
        type: string
        constraints: [ valid_values: [ 1111-2222-aaaa-bbbb ] ]
        default: 1111-2222-aaaa-bbbb
      flavour_id:
        type: string
        constraints: [ valid_values: [ simple ] ]
        default: simple

topology_template:
  substitution_mappings:
    node_type: tosca.example_NS
  requirements:
    virtual_link: [ VNF_2, virtual_link_2 ] # the External connection point of
# VNF_2 is exposed as the Sap

node_templates:
  my_service:
    type: tosca.example_NS
    properties:

```

```

descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
designer: Mycompany
name: ExampleService
version: '1.0'
invariant_id: 1111-2222-aaaa-bbbb
flavour_id: simple
interfaces:
  Nslcm:
    instantiate:
      implementation: instantiate.workflow.yaml
    terminate:
      implementation: terminate.workflow.yaml

VNF_1:
  type: toasca.nodes.nfv.example_VNF1
  properties:
    # no property assignments needed for required properties that have a default
value assigned in the node type definition, e.g. descriptor_id
    flavour_id: simple
    vnf_profile:
      instantiation_level: level_1
      min_number_of_instances: 2
      max_number_of_instances: 6
  requirements:
    - virtual_link: NsVirtualLink_1

VNF_2:
  type: toasca.nodes.nfv.example_VNF2
  properties:
    flavour_id: simple
    vnf_profile:
      instantiation_level: level_1
      min_number_of_instances: 1
      max_number_of_instances: 3
  requirements:
    - virtual_link_1: NsVirtualLink_1
    - virtual_link_2: # map to virtual_link requirement of the NS node

NsVirtualLink_1:
  type: toasca.nodes.nfv.NsVirtualLink
  properties:
    connectivity_type:
      layer_protocols: [ipv4]
      flow_pattern: mesh
    vl_profile:
      maxBitrateRequirements:
        root: 1000
      minBitrateRequirements:
        root: 1000

```

A.9 Mapping between NFV IM and TOSCA concepts

A.9.1 Introduction

This annex describes the mapping between the NFV information model and the TOSCA concepts.

A.9.2 Mapping between ETSI GS NFV-IFA 011 IM and TOSCA concepts

Table A.9.2-1 illustrates the mapping between the information model as specified in ETSI GS NFV-IFA 011 [1] and the corresponding TOSCA concepts.

Table A.9.2-1: Mapping between ETSI GS NFV-IFA 011 [1] IM and TOSCA concepts

ETSI GS NFV-IFA 014 [2] information element	TOSCA concept
VNFD (clause 7.1.2)	TOSCA service template(s) in the VNF package
VnfExtCpd (clause 7.1.4)	node template with type <code>tosca.nodes.nfv.VnfExtCp</code> or <code>tosca.nodes.nfv.VduCp</code>
VnfLcmOperationsConfiguration (clause 7.1.5)	property of VNF node type with data type <code>tosca.datatypes.nfv.VnfConfigurableProperties</code>
Vdu (clause 7.1.6)	node template with type <code>tosca.nodes.nfv.Vdu.Compute</code>
VLd (clause 7.1.7)	node template with type <code>tosca.nodes.nfv.VnfVirtualLink</code>
DeploymentFlavour (clause 7.1.8)	lower level service template(s) in the VNF package
VduProfile (clause 7.1.8.3)	property of Vdu.Compute node type with data type <code>tosca.datatypes.nfv.VduProfile</code>
VirtualLinkProfile (clause 7.1.8.4)	property of VnfVirtualLink node type with data type <code>tosca.datatypes.nfv.VIPProfile</code>
VirtualLinkDescFlavour (clause 7.1.8.5)	Only qos attribute in VirtualLinkDescFlavour has been defined as property of the VIPProfile data type
InstantiationLevel (clause 7.1.8.7)	policy with type <code>tosca.policies.nfv.InstantiationLevels</code>
VduLevel (clause 7.1.8.9)	policy with type <code>tosca.policies.nfv.VduInstantiationLevels</code>
LocalAffinityOrAntiAffinityRule (clause 7.1.8.11)	policy with type <code>tosca.policies.nfv.AffinityRule</code> or <code>tosca.policies.nfv.AntiAffinityRule</code>
AffinityOrAntiAffinityGroup (clause 7.1.8.12)	policy with type <code>tosca.policies.nfv.AffinityRule</code> or <code>tosca.policies.nfv.AntiAffinityRule</code>
VirtualComputeDesc (clause 7.1.9.2.2)	VirtualCompute capability of the Vdu.Compute node template
VirtualStorageDesc with BlockStorageData (clause 7.1.9.4.3)	node template with type <code>tosca.nodes.nfv.Vdu.VirtualBlockStorage</code>
VirtualStorageDesc with ObjectStorageData (clause 7.1.9.4.4)	node template with type <code>tosca.nodes.nfv.Vdu.VirtualObjectStorage</code>
VirtualStorageDesc with FileStorageData (clause 7.1.9.4.5)	node template with type <code>tosca.nodes.nfv.Vdu.VirtualFileStorage</code>
ScalingAspect (clause 7.1.10.2)	policy with type <code>tosca.policies.nfv.ScalingAspects</code>
ScalingDelta (clause 7.1.10.4)	policy with type <code>tosca.policies.nfv.VduScalingAspectDeltas</code> and/or <code>tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas</code>
VnfIndicator (clause 7.1.11.2)	Not supported in this version of the present document
MonitoringParameter (clause 7.1.11.3)	property of the VNF, Vdu.Compute and VnfVirtualLink node types
VnfConfigurableProperties (clause 7.1.12)	property of VNF node template with data type <code>tosca.datatypes.nfv.VnfConfigurableProperties</code>
LifeCycleManagementScript (clause 7.1.13)	interface with type <code>tosca.interfaces.nfv.Vnflcm</code>
VnfInfoModifiableAttributes (clause 7.1.14)	property of VNF node template with data type <code>tosca.datatypes.nfv.VnfInfoModifiableAttributes</code>

A.9.3 Mapping between IFA014 IM and TOSCA concepts

Table A.9.3-1 illustrates the mapping between the information model as specified in ETSI GS NFV-IFA 014 [2] and the corresponding TOSCA concepts.

Table A.9.3-1: Mapping between IFA014 IM and TOSCA concepts

ETSI GS NFV-IFA 014 [2] information element	TOSCA concept
NSD (clause 6.2.2)	TOSCA service template(s)
Sapd (clause 6.2.3)	node template with type <code>tosca.nodes.nfv.Sap</code> or
MonitoredData (clause 6.2.6)	Not supported in this version of the present document
VnfIndicatorData (clause 6.2.7)	Not supported in this version of the present document
LifeCycleManagementScript (clause 6.2.9)	Interface with type
NsDf (clause 6.3)	Only support single deployment flavour design by using a TOSCA service template
VnfProfile (clause 6.3.3)	property of VNF node template with data type <code>tosca.datatypes.nfv.VnfProfile</code>
VirtualLinkProfile (clause 6.3.4)	property of <code>NsVirtualLink</code> node template with data type <code>tosca.datatypes.nfv.VIPProfile</code>
AffinityOrAntiAffinityGroup (clause 6.3.5)	policy with type <code>tosca.policies.nfv.AffinityRule</code> or <code>tosca.policies.nfv.AntiAffinityRule</code>
NsVirtualLinkConnectivity (clause 6.3.7)	requirement and capability between <code>PnfExtCp</code> or <code>VnfExtCp</code> and <code>NsVirtualLink</code> node templates
NsLevel (clause 6.3.9)	Not supported in this version of the present document
NsScaleInfo (clause 6.3.10)	Not supported in this version of the present document
NsProfile (clause 6.3.11)	Not supported in this version of the present document
Dependencies (clause 6.3.12)	Not supported in this version of the present document
Vnffgd (clause 6.4.2)	Not supported in this version of the present document
Nfpd (clause 6.4.3)	Not supported in this version of the present document
CpdPool (clause 6.4.4)	Not supported in this version of the present document
NfpPositionDesc (clause 6.4.5)	Not supported in this version of the present document
CpProfile (clause 6.4.6)	Not supported in this version of the present document
NsVirtualLinkDesc (clause 6.5.2)	node template with type <code>tosca.nodes.nfv.NsVirtualLink</code>
VirtualLinkDf (clause 6.5.4)	Only the <code>serviceAvailabilityLevel</code> attribute in <code>VirtualLinkDf</code> has been defined as property of the <code>NsVirtualLink</code> node type
Pnfd (clause 6.6.2)	A TOSCA service template
PnfExtCpd (clause 6.6.4)	node template with type <code>tosca.nodes.nfv.PnfExtCp</code>
NsScalingAspect (clause 6.7.2)	Not supported in this version of the present document
VnfToLevelMapping (clause 6.7.4)	Not supported in this version of the present document
VirtualLinkToLevelMapping (clause 6.7.5)	Not supported in this version of the present document
NsToLevelMapping (clause 6.7.6)	Not supported in this version of the present document

A.10 PNFD modelling design example

In this example, TOSCA-Simple-Profile-yaml-v1.1 is used for designing and processing a PNFD TOSCA model.

The service template contains a node template of type `MyCompany.examplePnf.1_0` which represents the main part of the PNF and a node template of type `tosca.nodes.nfv.PnfExtCp` representing the PNF external connection point.

examplePnf.yaml

This example illustrates a PNF with one external connection point, `pnfExtCp_1`. All the rest parts of the PNF is described as a single box, e.g. called `pnf_mainPart` in the PNFD.

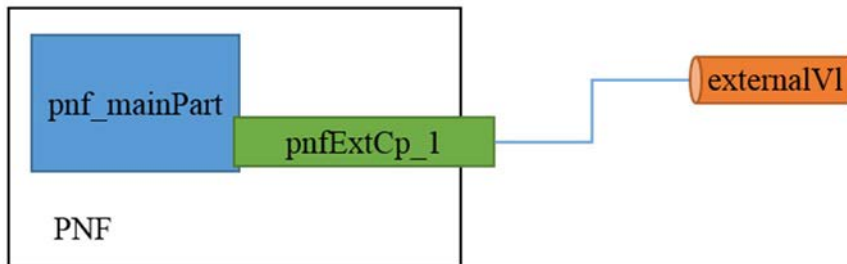


Figure A.10-1: examplePnf

```
tosca_definitions_version: tosca_simple_yaml_1_1
description: the service template of a PNFD
imports:
  - etsi_nfv_sol001_pnfd_2_5_1_types.yaml
node_types:
  MyCompany.examplePnf.1_0:
    derived_from: tosca.nodes.nfv.PNF
    properties:
      descriptor_id:
        type: string
        constraints: [ valid_values: [ b1bb0ce7-ebca-4fa7-95ed-4840d70a2233 ] ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233
      function_description:
        type: string
        default: an example PNF
    provider:
      type: string
      constraints: [ valid_values: [ MyCompany ] ]
      default: MyCompany
    version:
      type: string
      constraints: [ valid_values: [ '1.0' ] ]
      default: '1.0'
    descriptor_invariant_id:
      type: string
      constraints: [ valid_values: [ 1111-2222-ccaa-bbdd ] ]
      default: 1111-2222-ccaa-bbdd
    name:
      type: string
      constraints: [ valid_values: [ ExamplePnf ] ]
      default: ExamplePnf
    requirements:
      - virtual_link:
```

```
    capability: toasca.capabilities.nfv.VirtualLinkable

topology_template:
  substitution_mappings:
    node_type: MyCompany.examplePnf.1_0
  requirements:
    virtual_link: [ pnfExtCp_1, external_virtual_link ]

node_templates:
  pnf_mainPart:
    type: MyCompany.examplePnf.1_0
    properties:
      descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233
      function_description: an example PNF
      provider: Mycompany
      version: 1.0
      descriptor_invariant_id: 1111-2222-ccaa-bbdd
      name: ExamplePnf

  pnfExtCp_1:
    type: toasca.nodes.nfv.PnfExtCp
    properties:
      protocol: [ associated_layer_protocol: ipv4 ]
      trunk_mode: false
      layer_protocols: [ ipv4 ]
      role: leaf
      description: External connection point to access this pnf
    requirements:
      # - external_virtual_link:
      - dependency: pnf_mainPart
```

Annex B (normative): etsi_nfv_sol001_type definitions

B.1 Purpose

All type definitions specified in clauses 6, 7 and 8 of the present document are gathered in three definition files.

The file names are structured as follows:

- etsi_nfv_sol001_vnfd_x_y_z_types.yaml, for the definitions provided in clause 6;
- etsi_nfv_sol001_nsd_x_y_z_types.yaml, for the definitions provided in clause 7;
- etsi_nfv_sol001_pnfd_x_y_z_types.yaml, for the definitions provided in clause 8;

where x and y represents the version of this file and are set respectively to "2", "5" and "1" for this version of the present document.

In subsequent versions of the present document, "x", "y" and "z" in etsi_nfv_sol001_vnfd_x_y_z_types.yaml, etsi_nfv_sol001_nsd_x_y_z_types.yaml and etsi_nfv_sol001_pnfd_x_y_z_types.yaml will be incremented only if there are changes in the VNFD, NSD or PNFD type definitions, respectively. In that case, "x", "y" and "z" will have the same value as in the subsequent version of the present document where the changes are introduced.

B.2 VNFD type definitions file

The file etsi_nfv_sol001_vnfd_2_5_1_types.yaml, attached to the present document, contains all type definitions specified in clause 6 of the present document. This file is also available at the following URL:
https://forge.etsi.org/rep/nfv/SOL001/blob/master/etsi_nfv_sol001_vnfd_2_5_1_types.yaml.

A TOSCA service template representing a VNFD complying with the present document shall contain import statement referencing this file, as defined in clause 5.5.1.

NOTE: This file may, but need not, be included in the VNF Package.

B.3 NSD type definitions file

The file etsi_nfv_sol001_nsd_2_5_1_types.yaml, attached to the present document, contains all type definitions specified in clause 7 of the present document. This file is also available at the following URL:
https://forge.etsi.org/rep/nfv/SOL001/blob/master/etsi_nfv_sol001_nsd_2_5_1_types.yaml.

A TOSCA service template representing a NSD complying with the present document shall contain import statement referencing this file, as defined in clause 5.5.2.

NOTE: This file may, but need not, be included in the NSD file archive.

B.4 PNFD type definitions file

The file etsi_nfv_sol001_pnfd_2_5_1_types.yaml, attached to the present document, contains all type definitions specified in clause 8 of the present document. This file is also available at the following URL:
https://forge.etsi.org/rep/nfv/SOL001/blob/master/etsi_nfv_sol001_pnfd_2_5_1_types.yaml.

A TOSCA service template representing a PNFD complying with the present document shall contain import statement referencing this file, as defined in clause 5.5.3.

Annex C (normative): Conformance

C.1 Purpose

The present document specifies a data model for the VNFD, the NSD and the PNFD, by using the grammar defined in the TOSCA Simple Profile in YAML specification [3]. This annex specifies the requirements to be fulfilled for claiming conformance to the present document.

C.2 NFV TOSCA YAML service template

A VNFD, an NSD or a PNFD conforms to the present document if it complies with all the requirements below:

- 1) A VNFD conformant to the present document shall comply with the requirements in clause 6 of the present document and to the specification of the elements of the TOSCA Simple Profile in YAML specification [3] it uses, unless otherwise stated in clause 6 of the present document.
- 2) A NSD conformant to the present document shall comply with the requirements in clause 7 of the present document and to the specification of the elements of the TOSCA Simple Profile in YAML specification [3] it uses, unless otherwise stated in clause 7 of the present document.
- 3) A PNFD conformant to the present document shall comply with the requirements in clause 8 of the present document and to the specification of the elements of the TOSCA Simple Profile in YAML specification [3] it uses, unless otherwise stated in clause 8 of the present document.
- 4) When using or referring to the TOSCA normative types listed in table C.2-1, it is valid according to the definitions given in clause 6 and 7 of the present document and to section 5 of the TOSCA Simple Profile in YAML specification [3].

Table C.2-1: TOSCA normative types used in the present document

Types	VNFD	NSD
tosca.datatypes.Root	X	X
tosca.artifacts.File	X	
tosca.artifacts.Deployment.Image	X	
tosca.artifacts.Implementation	X	
tosca.capabilities.Root	X	
tosca.relationships.Root	X	
tosca.relationships.DependsOn	X	X
tosca.interfaces.Root	X	X
tosca.nodes.Root	X	X
tosca.groups.Root	X	X
tosca.policies.Root	X	X
tosca.policies.Placement	X	X

- 5) A VNFD conformant to the present document shall comply with VNFD TOSCA service template design specified in clause 6.11 of the present document.
- 6) A VNFD and NSD conformant to the present document shall comply with rules for Type extension defined in clause 5.6 of the present document.

C.3 NFV TOSCA processor

A processor or program conforms to the present document as NFV TOSCA processor for VNFD, NFV TOSCA processor for NSD or NFV TOSCA processor for PNFD if it complies with all the requirements below:

- 1) It can parse and recognize the elements of any VNFD, NSD or PNFD that conform to the present document, and shall generate errors for those documents that fail to conform to the present document.
- 2) It shall comply with all requirements and implement the semantics associated with the definitions specified in clauses 6, 7 and 8 of the present document.
- 3) It shall resolve the import definitions, either explicit or implicit, as described in clause 5.5 of the present document.

Annex D (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Shitao Li, Huawei

Other contributors:

Shitao Li, Huawei

Thinh Nguyenphu, Nokia

Andrei Kojukhov, Amdocs

Arturo Martin De Nicolas, Ericsson

Yuya Kuno, NTTDOCOMO

Gabor Marton, Nokia

Manchang Ju, ZTE

Bruno Chatras, Orange

Raquel Morera, Verizon

Satish Vasamsetti, Verizon Wireless

Philip Joseph, RIFT.io

Alex Vul, Intel

Priya T G, Netcracker Technologies Ltd

Li xiang, China Mobile US Research Cente

Ren rongwei, China Mobile US Research Cente

Yuan yannan, China Mobile US Research Cente

Uwe Rauschenbach, Nokia Germany

Nagesha Subramanya, Nokia Germany

kaliyaperumal Ramalingam. Verizon UK Ltd

Abhishek Kumar, Verizon UK Ltd

Philip Joseph, Riftio

Wangyaoguang, Huawei

Annex E (informative): Change History

Date	Version	Information about changes
2016.05	0.0.1	Implemented NFVSOL(16)00005r1_GS_NFV_SOL001_ToC
2016.07	0.02	Implemented NFVSOL(16)000026r1, NFVSOL(16)000027r1, NFVSOL(16)000028r2
2017.09	0.1.0	Implemented NFVSOL(17)000539r3, NFVSOL(17)000540r3, NFVSOL(17)000542r2, NFVSOL(17)000544r1
2017.10	0.2.0	Implemented NFVSOL(17)000543r3, NFVSOL(17)000566, NFVSOL(17)000567r1, NFVSOL(17)000568r1, NFVSOL(17)000569 Editorial modification for clause numbering and format.
2017.11	0.3.0	Clean-up done by <i>editHelp</i> Implemented NFVSOL(17)000545r4, NFVSOL(17)000559r2, NFVSOL(17)000560r2, NFVSOL(17)000570r1, NFVSOL(17)000575r1, NFVSOL(17)000616r1, NFVSOL(17)000641
2017.12	0.4.0	Implemented NFVSOL(17)000642r1, NFVSOL(17)000675r1, NFVSOL(17)000756
2018.01	0.5.0	Implemented NFVSOL(17)000621r6, NFVSOL(17)000676r3, NFVSOL(17)000677r2, NFVSOL(17)000736r1, NFVSOL(18)000004r2
2018.3	0.6.0	Implemented NFVSOL(18)000049, NFVSOL(18)000025r1, NFVSOL(18)00048R1, NFVSOL(18)00040, NFVSOL(18)00023r2, NFVSOL(18)00024r2, NFVSOL(18)00029r1, NFVSOL(18)00063r5, NFVSOL(18)000077, NFVSOL(18)000046r2, NFVSOL(18)000074r2, NFVSOL(17)0000611r5, NFVSOL(18)000041r1, NFVSOL(18)000038r3, NFVSOL(18)000028, NFVSOL(18)000044r2, NFVSOL(18)000055r2, NFVSOL(18)000045r1
2018.3	0.6.1	Implemented NFVSOL(18)000052r1, NFVSOL(18)000094r5, NFVSOL(18)000043r3, NFVSOL(18)0000112r1, NFVSOL(18)0000117r2, NFVSOL(18)0000124r1, NFVSOL(18)0000129r1, NFVSOL(18)000042r2, NFVSOL(18)0000115r2
2018.4	0.6.2	Implemented NFVSOL(18)000113r2, NFVSOL(18)000119r2, NFVSOL(18)000121, NFVSOL(18)000135, NFVSOL(18)000116r1, NFVSOL(18)000157, NFVSOL(18)000158r1
2018.5	0.6.3	Implemented NFVSOL(18)000168, NFVSOL(18)000169r2, NFVSOL(18)000173, NFVSOL(18)000174r2
2018.5	0.7.0	Implemented NFVSOL(18)000156r7, NFVSOL(18)000142r3, NFVSOL(18)000147, NFVSOL(18)000193r2, NFVSOL(18)000201, NFVSOL(18)000202, NFVSOL(18)000203r2, NFVSOL(18)000205r1, NFVSOL(18)000160r1, NFVSOL(18)000199r2, NFVSOL(18)000200r3, NFVSOL(18)000192r1, NFVSOL(18)000194r2, NFVSOL(18)000231r1, NFVSOL(18)000223r1, NFVSOL(18)000183r2
2018.6	0.8.0	Implemented NFVSOL(18)000287, NFVSOL(18)00012r11, NFVSOL(18)000197r2, NFVSOL(18)000198r3, NFVSOL(18)000286, NFVSOL(18)000292, NFVSOL(18)000294r1, NFVSOL(18)000295r1, NFVSOL(18)000301r2, NFVSOL(18)000302r2, NFVSOL(18)000240r1, NFVSOL(18)000253r3, NFVSOL(18)000254r2, NFVSOL(18)000256r2,
2018.6	0.9.0	Implemented NFVSOL(18)000288r2 Adding etsi_nfv_sol001_vnfd_0_9_0_type.yaml and SOL001 Graphics v0_9_0.pptx in the draft GS zip package Editorial changes for all the TOSCA type definitions.
2018.8	0.10.0	Implemented NFVSOL(18)000331r4, NFVSOL(18)000335r2, NFVSOL(18)000351r1, NFVSOL(18)000374r1, NFVSOL(18)000376, NFVSOL(18)000382, NFVSOL(18)000395r1, NFVSOL(18)000402r2, NFVSOL(18)000404, NFVSOL(18)000405r2, NFVSOL(18)000406r1, NFVSOL(18)000408r2, NFVSOL(18)000409r1, NFVSOL(18)000411r1, NFVSOL(18)000413, NFVSOL(18)000416r2, NFVSOL(18)000422r4, NFVSOL(18)000423r4, NFVSOL(18)000424r2, NFVSOL(18)000425r2, NFVSOL(18)000430r1, NFVSOL(18)000336r4, NFVSOL(18)000380, NFVSOL(18)000387r2, NFVSOL(18)000394r1, NFVSOL(18)000410r2, NFVSOL(18)000447r3, NFVSOL(18)000427, NFVSOL(18)000403r3, NFVSOL(18)000420r4, NFVSOL(18)000393r1, NFVSOL(18)000415, NFVSOL(18)000398r1, NFVSOL(18)000399r3, NFVSOL(18)000414r2, NFVSOL(18)000379r3, NFVSOL(18)000428r5, NFVSOL(18)000429r4, NFVSOL(18)000465r1, NFVSOL(18)000479, NFVSOL(18)000375r4, NFVSOL(18)000488r2, NFVSOL(18)000492r1

Date	Version	Information about changes
2018.9	0.11.0	Implemented NFVSOL(18)000486r7, NFVSOL(18)000495r1, NFVSOL(18)000497r3, NFVSOL(18)000498, NFVSOL(18)000500r3, NFVSOL(18)000503r1, NFVSOL(18)000504r1, NFVSOL(18)000505r1, NFVSOL(18)000508, NFVSOL(18)000514r2, NFVSOL(18)000515r1, NFVSOL(18)000516r1, NFVSOL(18)000524r1, NFVSOL(18)000529r2, NFVSOL(18)000530r1, NFVSOL(18)000536r1, NFVSOL(18)000538r1, NFVSOL(18)000541r3, NFVSOL(18)000544r1, NFVSOL(18)000545r1,
2018.10	0.12.0	Implemented NFVSOL(18)000507r3, NFVSOL(18)000621r4, NFVSOL(18)000547r1, NFVSOL(18)000567r1, NFVSOL(18)000574, NFVSOL(18)000579, NFVSOL(18)000590r1
2018.11	0.13.0	Implemented NFVSOL(18)000563r5, NFVSOL(18)000575r5, NFVSOL(18)000586r2, NFVSOL(18)000587r1, NFVSOL(18)000589r2, NFVSOL(18)000591, NFVSOL(18)000592, NFVSOL(18)000604r2, NFVSOL(18)000606r1, NFVSOL(18)000607r5, NFVSOL(18)000608, NFVSOL(18)000609, NFVSOL(18)000612, NFVSOL(18)000614r1, NFVSOL(18)000615, NFVSOL(18)000616r2, NFVSOL(18)000617r3, NFVSOL(18)000619r3, NFVSOL(18)000620r5, NFVSOL(18)000628r2, NFVSOL(18)000629r1, NFVSOL(18)000630, NFVSOL(18)000631r1, NFVSOL(18)000632, NFVSOL(18)000634, NFVSOL(18)000635, NFVSOL(18)000636r3, NFVSOL(18)000637, NFVSOL(18)000655r2, NFVSOL(18)000658r3, NFVSOL(18)000659, NFVSOL(18)000660, NFVSOL(18)000665r1, NFVSOL(18)000666, NFVSOL(18)000678r1, NFVSOL(18)000679, NFVSOL(18)000682, NFVSOL(18)000684r1, NFVSOL(18)000687

History

Document history		
V2.5.1	December 2018	Publication