ETSI GS NFV-SOL 001 V2.8.1 (2020-09)

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.

**Group Specification**

Reference

RGS/NFV-SOL001ed281

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 prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

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 2020.

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 19

Foreword 19

Modal verbs terminology 19

1 Scope 20

2 References 20

2.1 Normative references 20

2.2 Informative references 21

3 Definition of terms, symbols and abbreviations 22

3.1 Terms 22

3.2 Symbols 22

3.3 Abbreviations 22

4 Overview of TOSCA model 23

5 General concept of using TOSCA to model NFV descriptors 23

5.1 Introduction 23

5.2 Network Service Descriptor 23

5.3 Virtualised Network Function Descriptor 23

5.4 Physical Network Function Descriptor 24

5.5 tosca\_definitions\_version and Namespace prefix 24

5.6 Imports statement 25

5.6.1 VNFD TOSCA service template 25

5.6.2 NSD TOSCA service template 25

5.6.3 PNFD TOSCA service template 26

5.7 Type extension 26

5.7.1 Introduction 26

5.7.2 Rules 26

5.7.3 VNFD Types 27

5.7.4 NSD types 28

5.8 Non-Backward Compatible changes 28

6 VNFD TOSCA model 28

6.1 Introduction 28

6.2 Data Types 29

6.2.1 tosca.datatypes.nfv.CpProtocolData 29

6.2.1.1 Description 29

6.2.2 tosca.datatypes.nfv.AddressData 30

6.2.2.1 Description 30

6.2.3 tosca.datatypes.nfv.L2AddressData 30

6.2.3.1 Description 30

6.2.4 tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements 30

6.2.4.1 Description 30

6.2.4.2 Properties 30

6.2.4.3 Definition 31

6.2.4.4 Examples 31

6.2.4.5 Additional Requirements 31

6.2.5 tosca.datatypes.nfv.L3AddressData 31

6.2.5.1 Description 31

6.2.6 tosca.datatypes.nfv.RequestedAdditionalCapability 32

6.2.6.1 Description 32

6.2.6.2 Properties 32

6.2.6.3 Definition 33

6.2.6.4 Examples 33

6.2.6.5 Additional Requirements 33

6.2.7 tosca.datatypes.nfv.VirtualMemory 33

6.2.7.1 Description 33

6.2.7.2 Properties 34

6.2.7.3 Definitions 34

6.2.7.4 Examples 34

6.2.7.5 Additional Requirements 35

6.2.8 tosca.datatypes.nfv.VirtualCpu 35

6.2.8.1 Description 35

6.2.8.2 Properties 35

6.2.8.3 Definition 35

6.2.8.4 Examples 36

6.2.8.5 Additional Requirements 36

6.2.9 tosca.datatypes.nfv.VirtualCpuPinning 36

6.2.9.1 Description 36

6.2.9.2 Properties 36

6.2.9.3 Definition 37

6.2.9.4 Examples 37

6.2.9.5 Additional Requirements 37

6.2.10 tosca.datatypes.nfv.VnfcConfigurableProperties 37

6.2.10.1 Description 37

6.2.10.2 Properties 38

6.2.10.3 Definition 38

6.2.10.4 Examples 38

6.2.10.5 Additional Requirements 39

6.2.11 tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties 39

6.2.11.1 Description 39

6.2.11.2 Properties 39

6.2.11.3 Definition 39

6.2.12 tosca.datatypes.nfv.VduProfile 39

6.2.12.1 Description 39

6.2.12.2 Properties 40

6.2.12.3 Definition 40

6.2.12.4 Examples 41

6.2.12.5 Additional requirements 42

6.2.13 tosca.datatypes.nfv.VlProfile 42

6.2.13.1 Description 42

6.2.13.2 Properties 42

6.2.13.3 Definition 43

6.2.13.4 Examples 43

6.2.13.5 Additional Requirements 43

6.2.14 tosca.datatypes.nfv.VirtualLinkProtocolData 43

6.2.14.1 Description 43

6.2.14.2 Properties 44

6.2.14.3 Definition 44

6.2.14.4 Examples 44

6.2.14.5 Additional Requirements 44

6.2.15 tosca.datatypes.nfv.L2ProtocolData 45

6.2.15.1 Description 45

6.2.15.2 Properties 45

6.2.15.3 Definition 45

6.2.15.4 Examples 46

6.2.15.5 Additional Requirements 46

6.2.16 tosca.datatypes.nfv.L3ProtocolData 46

6.2.16.1 Description 46

6.2.16.2 Properties 46

6.2.16.3 Definition 47

6.2.16.4 Examples 48

6.2.16.5 Additional Requirements 48

6.2.17 tosca.datatypes.nfv.IpAllocationPool 48

6.2.17.1 Description 48

6.2.17.2 Properties 48

6.2.17.3 Definition 49

6.2.17.4 Examples 49

6.2.17.5 Additional Requirements 49

6.2.18 tosca.datatypes.nfv.InstantiationLevel 49

6.2.18.1 Description 49

6.2.18.2 Properties 49

6.2.18.3 Definition 50

6.2.18.4 Examples 50

6.2.18.5 Additional Requirements 50

6.2.19 tosca.datatypes.nfv.VduLevel 50

6.2.19.1 Description 50

6.2.19.2 Properties 50

6.2.19.3 Definition 51

6.2.19.4 Examples 51

6.2.19.5 Additional Requirements 51

6.2.20 tosca.datatypes.nfv.VnfLcmOperationsConfiguration 51

6.2.20.1 Description 51

6.2.20.2 Properties 51

6.2.20.3 Definition 52

6.2.20.4 Examples 53

6.2.20.5 Additional Requirements 53

6.2.21 tosca.datatypes.nfv.VnfInstantiateOperationConfiguration 53

6.2.21.1 Description 53

6.2.21.2 Properties 53

6.2.21.3 Definition 53

6.2.21.4 Examples 53

6.2.21.5 Additional Requirements 53

6.2.22 tosca.datatypes.nfv.VnfScaleOperationConfiguration 54

6.2.22.1 Description 54

6.2.22.2 Properties 54

6.2.22.3 Definition 54

6.2.22.4 Examples 54

6.2.22.5 Additional Requirements 54

6.2.23 tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration 55

6.2.23.1 Description 55

6.2.23.2 Properties 55

6.2.23.3 Definition 55

6.2.23.4 Examples 55

6.2.23.5 Additional Requirements 55

6.2.24 tosca.datatypes.nfv.VnfHealOperationConfiguration 55

6.2.24.1 Description 55

6.2.24.2 Properties 56

6.2.24.3 Definition 56

6.2.24.4 Examples 56

6.2.24.5 Additional Requirements 56

6.2.25 tosca.datatypes.nfv.VnfTerminateOperationConfiguration 56

6.2.25.1 Description 56

6.2.25.2 Properties 57

6.2.25.3 Definition 57

6.2.25.4 Examples 57

6.2.25.5 Additional Requirements 57

6.2.26 tosca.datatypes.nfv.VnfOperateOperationConfiguration 57

6.2.26.1 Description 57

6.2.26.2 Properties 58

6.2.26.3 Definition 58

6.2.26.4 Examples 58

6.2.26.5 Additional Requirements 58

6.2.27 tosca.datatypes.nfv.ScaleInfo 59

6.2.27.1 Description 59

6.2.27.2 Properties 59

6.2.27.3 Definition 59

6.2.27.4 Examples 59

6.2.27.5 Additional Requirements 59

6.2.28 tosca.datatypes.nfv.ScalingAspect 59

6.2.28.1 Description 59

6.2.28.2 Properties 60

6.2.28.3 Definition 60

6.2.28.4 Examples 61

6.2.28.5 Additional Requirements 61

6.2.29 tosca.datatypes.nfv.LinkBitrateRequirements 61

6.2.29.1 Description 61

6.2.30 tosca.datatypes.nfv.ConnectivityType 61

6.2.30.1 Description 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 63

6.2.31.4 Examples 64

6.2.31.5 Additional Requirements 65

6.2.32 tosca.datatypes.nfv.VnfAdditionalConfigurableProperties 66

6.2.32.1 Description 66

6.2.32.2 Properties 66

6.2.32.3 Definition 66

6.2.32.4 Examples 66

6.2.32.5 Additional Requirements 67

6.2.33 tosca.datatypes.nfv.VnfInfoModifiableAttributes 67

6.2.33.1 Description 67

6.2.33.2 Properties 67

6.2.33.3 Definition 67

6.2.33.4 Examples 67

6.2.33.5 Additional Requirements 68

6.2.34 tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions 68

6.2.34.1 Description 68

6.2.34.2 Properties 68

6.2.34.3 Definition 68

6.2.34.4 Examples 68

6.2.34.5 Additional Requirements 68

6.2.35 tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata 68

6.2.35.1 Description 68

6.2.35.2 Properties 68

6.2.35.3 Definition 69

6.2.35.4 Examples 69

6.2.35.5 Additional Requirements 69

6.2.36 tosca.datatypes.nfv.Qos 69

6.2.36.1 Description 69

6.2.37 tosca.datatypes.nfv.LogicalNodeData 70

6.2.37.1 Description 70

6.2.37.2 Properties 70

6.2.37.3 Definition 70

6.2.37.4 Examples 70

6.2.37.5 Additional Requirements 71

6.2.38 tosca.datatypes.nfv.SwImageData 71

6.2.38.1 Description 71

6.2.38.2 Properties 71

6.2.38.3 Definition 72

6.2.38.4 Examples 73

6.2.38.5 Additional Requirements 73

6.2.39 tosca.datatypes.nfv.VirtualBlockStorageData 73

6.2.39.1 Description 73

6.2.39.2 Properties 74

6.2.39.3 Definition 74

6.2.39.4 Examples 74

6.2.39.5 Additional Requirements 75

6.2.40 tosca.datatypes.nfv.VirtualObjectStorageData 75

6.2.40.1 Description 75

6.2.40.2 Properties 75

6.2.40.3 Definition 75

6.2.40.4 Examples 75

6.2.40.5 Additional Requirements 75

6.2.41 tosca.datatypes.nfv.VirtualFileStorageData 76

6.2.41.1 Description 76

6.2.41.2 Properties 76

6.2.41.3 Definition 76

6.2.41.4 Examples 76

6.2.41.5 Additional Requirements 76

6.2.42 tosca.datatypes.nfv.VirtualLinkBitrateLevel 77

6.2.42.1 Description 77

6.2.42.2 Properties 77

6.2.42.3 Definition 77

6.2.42.4 Examples 77

6.2.42.5 Additional Requirements 77

6.2.43 tosca.datatypes.nfv.VnfOperationAdditionalParameters 77

6.2.43.1 Description 77

6.2.43.2 Properties 78

6.2.43.3 Definition 78

6.2.43.4 Examples 78

6.2.43.5 Additional Requirements 78

6.2.44 tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration 79

6.2.44.1 Description 79

6.2.44.2 Properties 79

6.2.44.3 Definition 79

6.2.44.4 Examples 79

6.2.44.5 Additional Requirements 79

6.2.45 tosca.datatypes.nfv.VnfChangeExtConnectivityOperation Configuration 79

6.2.45.1 Description 79

6.2.45.2 Properties 80

6.2.45.3 Definition 80

6.2.45.4 Examples 80

6.2.45.5 Additional Requirements 80

6.2.46 tosca.datatypes.nfv.VnfMonitoringParameter 80

6.2.47 tosca.datatypes.nfv.VnfcMonitoringParameter 80

6.2.47.1 Description 80

6.2.47.2 Properties 80

6.2.47.3 Definition 82

6.2.47.4 Examples 82

6.2.47.5 Additional Requirements 82

6.2.48 tosca.datatypes.nfv.VirtualLinkMonitoringParameter 82

6.2.48.1 Description 82

6.2.48.2 Properties 83

6.2.48.3 Definition 83

6.2.48.4 Examples 83

6.2.48.5 Additional Requirements 84

6.2.49 tosca.datatypes.nfv.InterfaceDetails 84

6.2.49.1 Description 84

6.2.49.2 Properties 84

6.2.49.3 Definition 84

6.2.49.4 Examples 84

6.2.49.5 Additional Requirements 85

6.2.50 tosca.datatypes.nfv.UriComponents 85

6.2.50.1 Description 85

6.2.50.2 Properties 85

6.2.50.3 Definition 85

6.2.50.4 Examples 86

6.2.50.5 Additional Requirements 86

6.2.51 tosca.datatypes.nfv.UriAuthority 86

6.2.51.1 Description 86

6.2.51.2 Properties 86

6.2.51.3 Definition 87

6.2.51.4 Examples 87

6.2.51.5 Additional Requirements 87

6.2.52 tosca.datatypes.nfv.VnfProfile 87

6.2.52.1 Description 87

6.2.53 tosca.datatypes.nfv.ChecksumData 88

6.2.53.1 Description 88

6.2.53.2 Properties 88

6.2.53.3 Definition 88

6.2.53.4 Examples 88

6.2.53.5 Additional Requirements 89

6.2.54 tosca.datatypes.nfv.VnfmInterfaceInfo 89

6.2.54.1 Description 89

6.2.54.2 Properties 89

6.2.54.3 Definition 89

6.2.54.4 Examples 90

6.2.54.5 Additional Requirements 90

6.2.55 tosca.datatypes.nfv.OauthServerInfo 90

6.2.55.1 Description 90

6.2.55.2 Properties 90

6.2.55.3 Definition 90

6.2.55.4 Examples 90

6.2.55.5 Additional Requirements 90

6.2.56 tosca.datatypes.nfv.BootData 91

6.2.56.1 Description 91

6.2.56.2 Properties 91

6.2.56.3 Definition 92

6.2.56.4 Examples 92

6.2.56.5 Additional Requirements 92

6.2.57 tosca.datatypes.nfv.KvpData 92

6.2.57.1 Description 92

6.2.57.2 Properties 93

6.2.57.3 Definition 93

6.2.57.4 Examples 93

6.2.57.5 Additional Requirements 93

6.2.58 tosca.datatypes.nfv.ContentOrFileData 93

6.2.58.1 Description 93

6.2.58.2 Properties 93

6.2.58.3 Definition 94

6.2.58.4 Examples 95

6.2.58.5 Additional Requirements 95

6.2.59 tosca.datatypes.nfv.BootDataVimSpecificProperties 95

6.2.59.1 Description 95

6.2.59.2 Properties 95

6.2.59.3 Definition 96

6.2.59.4 Examples 96

6.2.59.5 Additional Requirements 96

6.3 Artifact Types 96

6.3.1 tosca.artifacts.nfv.SwImage 96

6.3.1.1 Description 96

6.3.1.2 Description 96

6.3.2 tosca.artifacts.Implementation.nfv.Mistral 97

6.3.2.1 Description 97

6.3.2.2 Definition 97

6.4 Capability Types 97

6.4.1 tosca.capabilities.nfv.VirtualBindable 97

6.4.1.1 Description 97

6.4.1.2 Properties 97

6.4.1.3 Definition 97

6.4.2 tosca.capabilities.nfv.VirtualLinkable 98

6.4.2.1 Description 98

6.4.3 tosca.capabilities.nfv.VirtualCompute 98

6.4.3.1 Description 98

6.4.3.2 Properties 98

6.4.3.3 Definition 98

6.4.4 tosca.capabilities.nfv.VirtualStorage 99

6.4.4.1 Description 99

6.4.4.2 Definition 99

6.5 Requirements Types 99

6.6 Relationship Types 100

6.6.1 tosca.relationships.nfv.VirtualBindsTo 100

6.6.1.1 Description 100

6.6.1.2 Properties 100

6.6.1.3 Definition 100

6.6.2 tosca.relationships.nfv.VirtualLinksTo 100

6.6.2.1 Description 100

6.6.3 tosca.relationships.nfv.AttachesTo 100

6.6.3.1 Description 100

6.6.3.2 Properties 100

6.6.3.3 Definition 101

6.7 Interface Types 101

6.7.1 tosca.interfaces.nfv.Vnflcm 101

6.7.1.1 Description 101

6.7.1.2 Definition 102

6.7.1.3 Additional Requirements 103

6.7.1.4 Support of LCM scripts 104

6.7.1.5 Examples 105

6.7.2 tosca.interfaces.nfv.VnfIndicator 107

6.7.2.1 Description 107

6.7.2.2 Definition 107

6.7.2.3 Examples 107

6.8 Node Types 108

6.8.1 tosca.nodes.nfv.VNF 108

6.8.1.1 Description 108

6.8.1.2 Properties 108

6.8.1.3 Attributes 110

6.8.1.4 Requirements 110

6.8.1.5 Capabilities 110

6.8.1.6 Definition 110

6.8.1.7 Artifact 112

6.8.1.8 Additional Requirements 112

6.8.1.9 Example 114

6.8.2 tosca.nodes.nfv.VnfExtCp 116

6.8.2.1 Description 116

6.8.2.2 Properties 117

6.8.2.3 Attributes 117

6.8.2.4 Requirements 117

6.8.2.5 Capabilities 117

6.8.2.6 Definition 117

6.8.2.7 Additional Requirements 118

6.8.2.8 Example 118

6.8.3 tosca.nodes.nfv.Vdu.Compute 118

6.8.3.1 Description 118

6.8.3.2 Properties 118

6.8.3.3 Attributes 119

6.8.3.4 Requirements 119

6.8.3.5 Capabilities 120

6.8.3.6 Definition 120

6.8.3.7 Additional requirements 121

6.8.3.8 Example 122

6.8.4 tosca.nodes.nfv.Vdu.VirtualBlockStorage 126

6.8.4.1 Description 126

6.8.4.2 Properties 126

6.8.4.3 Attributes 127

6.8.4.4 Requirements 127

6.8.4.5 Capabilities 127

6.8.4.6 Definition 127

6.8.4.7 Additional requirements 127

6.8.5 tosca.nodes.nfv.Vdu.VirtualObjectStorage 127

6.8.5.1 Description 127

6.8.5.2 Properties 128

6.8.5.3 Attributes 128

6.8.5.4 Requirements 128

6.8.5.5 Capabilities 128

6.8.5.6 Definition 128

6.8.5.7 Additional requirements 128

6.8.6 tosca.nodes.nfv.Vdu.VirtualFileStorage 129

6.8.6.1 Description 129

6.8.6.2 Properties 129

6.8.6.3 Attributes 129

6.8.6.4 Requirements 129

6.8.6.5 Capabilities 129

6.8.6.6 Definition 130

6.8.6.7 Additional requirements 130

6.8.7 tosca.nodes.nfv.Cp 130

6.8.7.1 Description 130

6.8.8 tosca.nodes.nfv.VduCp 130

6.8.8.1 Description 130

6.8.8.2 Properties 130

6.8.8.3 Attributes 131

6.8.8.4 Requirements 131

6.8.8.5 Definition 132

6.8.9 tosca.nodes.nfv.VnfVirtualLink 132

6.8.9.1 Description 132

6.8.9.2 Properties 133

6.8.9.3 Requirements 133

6.8.9.4 Capabilities 133

6.8.9.5 Definition 133

6.8.10 tosca.nodes.nfv.VipCp 134

6.8.10.1 Description 134

6.8.10.2 Properties 134

6.8.10.3 Attributes 134

6.8.10.4 Requirements 135

6.8.10.5 Definition 135

6.8.10.6 Example 135

6.9 Group Types 135

6.9.1 tosca.groups.nfv.PlacementGroup 135

6.9.1.1 Description 135

6.9.1.2 Properties 136

6.9.1.3 Definition 136

6.9.1.4 Additional Requirements 136

6.9.1.5 Examples 136

6.10 Policy Types 136

6.10.1 tosca.policies.nfv.InstantiationLevels 136

6.10.1.1 Description 136

6.10.1.2 Properties 137

6.10.1.3 Definition 137

6.10.2 tosca.policies.nfv.VduInstantiationLevels 137

6.10.2.1 Description 137

6.10.2.2 Properties 138

6.10.2.3 Definition 138

6.10.2.4 Additional Requirements 138

6.10.3 tosca.policies.nfv.VirtualLinkInstantiationLevels 138

6.10.3.1 Description 138

6.10.3.2 Properties 139

6.10.3.3 Definition 139

6.10.3.4 Additional Requirements 139

6.10.4 Void 139

6.10.5 tosca.policies.nfv.ScalingAspects 139

6.10.5.1 Description 139

6.10.5.2 Properties 140

6.10.5.3 Definition 140

6.10.5.4 Examples 140

6.10.6 tosca.policies.nfv.VduScalingAspectDeltas 140

6.10.6.1 Description 140

6.10.6.2 Properties 141

6.10.6.3 Definition 141

6.10.6.4 Examples 141

6.10.7 tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas 141

6.10.7.1 Description 141

6.10.7.2 Properties 142

6.10.7.3 Definition 142

6.10.7.4 Examples 142

6.10.8 tosca.policies.nfv.VduInitialDelta 142

6.10.8.1 Description 142

6.10.8.2 Properties 143

6.10.8.3 Definition 143

6.10.8.4 Examples 143

6.10.9 tosca.policies.nfv.VirtualLinkBitrateInitialDelta 143

6.10.9.1 Description 143

6.10.9.2 Properties 143

6.10.9.3 Definition 144

6.10.9.4 Examples 144

6.10.10 AffinityRule, AntiAffinityRule 144

6.10.10.1 Description 144

6.10.10.2 Properties 144

6.10.10.3 targets 145

6.10.10.4 Definition 145

6.10.10.5 Examples 146

6.10.11 tosca.policies.nfv.Abstract.SecurityGroupRule 146

6.10.11.1 Description 146

6.10.12 tosca.policies.nfv.SupportedVnfInterface 146

6.10.12.1 Description 146

6.10.12.2 Properties 147

6.10.12.3 Definition 147

6.10.12.4 Additional requirements 147

6.10.12.5 Example 147

6.10.13 tosca.policies.nfv.SecurityGroupRule 148

6.10.13.1 Description 148

6.10.13.2 Properties 148

6.10.13.3 targets 148

6.10.13.4 Definition 148

6.10.13.5 Additional Requirements 148

6.11 VNFD TOSCA service template design 149

6.11.1 General 149

6.11.2 Single or multiple deployment flavour design with TOSCA-Simple-Profile-YAML-v1.2 149

6.11.3 Single deployment flavour design with TOSCA-Simple-Profile-YAML-v1.1 150

7 NSD TOSCA model 151

7.1 Introduction 151

7.2 Data Types 151

7.2.1 tosca.datatype.nfv.ServiceAvailability 151

7.2.1.1 Description 151

7.2.1.2 Properties 151

7.2.1.3 Definition 152

7.2.1.4 Examples 152

7.2.2 tosca.datatypes.nfv.VnfProfile 152

7.2.2.1 Description 152

7.2.3 tosca.datatype.nfv.NsVlProfile 152

7.2.3.1 Description 152

7.2.3.2 Properties 152

7.2.3.3 Definition 153

7.2.3.4 Examples 153

7.2.3.5 Additional Requirements 153

7.2.4 tosca.datatypes.nfv.ConnectivityType 154

7.2.4.1 Description 154

7.2.5 tosca.datatypes.nfv.NsVirtualLinkQos 154

7.2.5.1 Description 154

7.2.5.2 Properties 154

7.2.5.3 Definition 154

7.2.5.4 Examples 154

7.2.5.5 Additional Requirements 154

7.2.6 tosca.datatypes.nfv.LinkBitrateRequirements 155

7.2.6.1 Description 155

7.2.7 Void 155

7.2.8 Void 155

7.2.9 Void 155

7.2.10 Void 155

7.2.11 tosca.datatypes.nfv.CpProtocolData 155

7.2.11.1 Description 155

7.2.12 tosca.datatypes.nfv.AddressData 155

7.2.12.1 Description 155

7.2.13 tosca.datatypes.nfv.L2AddressData 155

7.2.13.1 Description 155

7.2.14 tosca.datatypes.nfv.L3AddressData 155

7.2.14.1 Description 155

7.2.15 tosca.datatypes.nfv.Qos 155

7.2.15.1 Description 155

7.2.16 tosca.datatypes.nfv.NsProfile 155

7.2.16.1 Description 155

7.2.16.2 Properties 156

7.2.16.3 Definition 156

7.2.16.4 Example 157

7.2.16.5 Additional Requirements 157

7.2.17 tosca.datatypes.nfv.Mask 157

7.2.17.1 Description 157

7.2.17.2 Properties 157

7.2.17.3 Definition 157

7.2.17.4 Examples 158

7.2.18 tosca.datatypes.nfv.NsOperationAdditionalParameters 158

7.2.18.1 Description 158

7.2.18.2 Properties 158

7.2.18.3 Definition 158

7.2.18.4 Examples 158

7.2.19 tosca.datatypes.nfv.NsMonitoringParameter 159

7.2.19.1 Description 159

7.2.19.2 Properties 159

7.2.19.3 Definition 160

7.2.19.4 Examples 160

7.2.19.5 Additional Requirements 160

7.2.20 tosca.datatypes.nfv.VnfMonitoringParameter 161

7.3 Artifact Types 161

7.4 Capabilities Types 161

7.4.1 tosca.capabilities.nfv.VirtualLinkable 161

7.4.1.1 Description 161

7.4.2 tosca.capabilities.nfv.Forwarding 161

7.4.2.1 Description 161

7.4.2.2 Properties 161

7.4.2.3 Definition 161

7.5 Requirements Types 161

7.6 Relationship Types 162

7.6.1 tosca.relationships.nfv.VirtualLinksTo 162

7.6.1.1 Description 162

7.6.2 tosca.relationships.nfv.ForwardTo 162

7.6.2.1 Description 162

7.6.2.2 Properties 162

7.6.2.3 Definition 162

7.7 Interface Types 162

7.7.1 tosca.interfaces.nfv.Nslcm 162

7.7.1.1 Description 162

7.7.1.2 Definition 163

7.7.1.3 Additional Requirements 164

7.7.1.4 Support of LCM scripts 164

7.7.1.5 Examples 165

7.8 Node Types 165

7.8.1 tosca.nodes.nfv.NS 165

7.8.1.1 Description 165

7.8.1.2 Properties 166

7.8.1.3 Attributes 166

7.8.1.4 Requirements 166

7.8.1.5 Capabilities 166

7.8.1.6 Definition 166

7.8.1.7 Artifact 167

7.8.1.8 Additional requirements 167

7.8.2 tosca.nodes.nfv.Sap 168

7.8.2.1 Description 168

7.8.2.2 Properties 168

7.8.2.3 Attributes 168

7.8.2.4 Requirements 168

7.8.2.5 Capabilities 168

7.8.2.6 Definition 168

7.8.2.7 Additional requirements 169

7.8.2.8 Example 169

7.8.3 tosca.nodes.nfv.NsVirtualLink 169

7.8.3.1 Description 169

7.8.3.2 Properties 169

7.8.3.3 Attributes 170

7.8.3.4 Requirements 170

7.8.3.5 Capabilities 170

7.8.3.6 Definition 170

7.8.3.7 Artifact 170

7.8.3.8 Additional Requirements 171

7.8.3.9 Example 171

7.8.4 tosca.nodes.nfv.Cp 171

7.8.4.1 Description 171

7.8.5 tosca.nodes.nfv.NfpPositionElement 171

7.8.5.1 Description 171

7.8.5.2 Properties 171

7.8.5.3 Attributes 171

7.8.5.4 Requirements 171

7.8.5.5 Capabilities 172

7.8.5.6 Definition 172

7.8.5.7 Artifact 172

7.8.5.8 Additional Requirements 172

7.8.5.9 Example 172

7.8.6 tosca.nodes.nfv.NFP 172

7.8.6.1 Description 172

7.8.6.2 Properties 173

7.8.6.3 Attributes 173

7.8.6.4 Requirements 173

7.8.6.5 Capabilities 173

7.8.6.6 Definition 173

7.8.7 tosca.nodes.nfv.NfpPosition 173

7.8.7.1 Description 173

7.8.7.2 Properties 173

7.8.7.3 Attributes 174

7.8.7.4 Requirements 174

7.8.7.5 Capabilities 174

7.8.7.6 Definition 175

7.8.7.7 Artifact 175

7.8.7.8 Additional Requirements 175

7.8.7.9 Example 175

7.8.8 tosca.nodes.nfv.Forwarding 175

7.8.8.1 Description 175

7.8.8.2 Properties 176

7.8.8.3 Attributes 176

7.8.8.4 Requirements 176

7.8.8.5 Capabilities 176

7.8.5.6 Definition 176

7.8.8.7 Artifact 177

7.8.8.8 Additional Requirements 177

7.8.8.9 Example 177

7.9 Group Types 177

7.9.1 tosca.groups.nfv.NsPlacementGroup 177

7.9.1.1 Description 177

7.9.1.2 Properties 177

7.9.1.3 Definition 177

7.9.1.4 Additional Requirements 178

7.9.2 tosca.groups.nfv.VNFFG 178

7.9.2.1 Description 178

7.9.2.2 Properties 178

7.9.2.3 Definition 178

7.9.2.4 Additional Requirements 178

7.9.2.5 Example 178

7.10 Policy Types 179

7.10.1 NsAffinityRule, NsAntiAffinityRule 179

7.10.1.1 Description 179

7.10.1.2 Properties 179

7.10.1.3 Targets 179

7.10.1.4 Definition 180

7.10.1.5 Examples 180

7.10.2 tosca.policies.nfv.NsSecurityGroupRule 181

7.10.2.1 Description 181

7.10.2.2 Properties 181

7.10.2.3 targets 181

7.10.2.4 Definition 182

7.10.2.5 Additional Requirements 182

7.10.3 tosca.policies.nfv.NfpRule 182

7.10.3.1 Description 182

7.10.3.2 Properties 182

7.10.3.3 Targets 183

7.10.3.4 Definition 183

7.10.3.5 Example 184

7.10.4 tosca.policies.nfv.NsMonitoring 184

7.10.4.1 Description 184

7.10.4.2 Properties 185

7.10.4.3 targets 185

7.10.4.4 Definition 185

7.10.4.5 Additional Requirements 185

7.10.5 tosca.policies.nfv.VnfMonitoring 186

7.10.5.1 Description 186

7.10.5.2 Properties 186

7.10.5.3 targets 186

7.10.5.4 Definition 186

7.10.5.5 Additional Requirements 187

7.10.6 tosca.policies.nfv.Abstract.SecurityGroupRule 187

7.10.6.1 Description 187

7.11 NSD TOSCA service template design 187

7.11.1 General 187

7.11.2 Single or multiple deployment flavour design with TOSCA-Simple-Profile-YAML-v1.2 187

7.11.3 Single deployment flavour design with TOSCA-Simple-Profile-YAML-v1.1 188

8 PNFD TOSCA model 189

8.1 Introduction 189

8.2 Data Types 190

8.2.1 tosca.datatypes.nfv.CpProtocolData 190

8.2.1.1 Description 190

8.2.2 tosca.datatypes.nfv.AddressData 190

8.2.2.1 Description 190

8.2.3 tosca.datatypes.nfv.L2AddressData 190

8.2.3.1 Description 190

8.2.4 tosca.datatypes.nfv.L3AddressData 190

8.2.4.1 Description 190

8.2.5 tosca.datatypes.nfv.LocationInfo 191

8.2.5.1 Description 191

8.2.5.2 Properties 191

8.2.5.3 Definition 191

8.2.5.4 Examples 192

8.2.5.5 Additional Requirements 192

8.2.6 tosca.datatypes.nfv.CivicAddressElement 192

8.2.6.1 Description 192

8.2.6.2 Properties 192

8.2.6.3 Definition 192

8.2.6.4 Examples 193

8.2.6.5 Additional Requirements 193

8.2.7 tosca.datatypes.nfv.GeographicCoordinates 193

8.2.7.1 Description 193

8.2.7.2 Properties 193

8.2.7.3 Definition 194

8.2.7.4 Examples 194

8.2.7.5 Additional Requirements 194

8.3 Artifact Types 194

8.4 Capabilities Types 195

8.4.1 tosca.capabilities.nfv.VirtualLinkable 195

8.4.1.1 Description 195

8.5 Requirements Types 195

8.6 Relationship Types 195

8.6.1 tosca.relationships.nfv.VirtualLinksTo 195

8.6.1.1 Description 195

8.7 Interface Types 195

8.8 Node Types 195

8.8.1 tosca.nodes.nfv.PNF 195

8.8.1.1 Description 195

8.8.1.2 Properties 195

8.8.1.3 Attributes 196

8.8.1.4 Requirements 196

8.8.1.5 Capabilities 196

8.8.1.6 Definition 196

8.8.1.7 Artifact 197

8.8.1.8 Additional Requirements 197

8.8.1.9 Example 197

8.8.2 tosca.nodes.nfv.PnfExtCp 197

8.8.2.1 Description 197

8.8.2.2 Properties 198

8.8.2.3 Attributes 198

8.8.2.4 Requirements 198

8.8.2.5 Capabilities 198

8.8.2.6 Definition 198

8.8.3 tosca.nodes.nfv.Cp 198

8.8.3.1 Description 198

8.9 Group Types 198

8.10 Policy Types 199

8.10.1 tosca.policies.nfv.PnfSecurityGroupRule 199

8.10.1.1 Description 199

8.10.1.2 Properties 199

8.10.1.3 targets 199

8.10.1.4 Definition 199

8.10.1.5 Additional Requirements 199

8.10.2 tosca.policies.nfv.Abstract.SecurityGroupRule 199

8.10.2.1 Description 199

8.11 PNFD TOSCA service template design 200

8.11.1 General 200

9 Common Definitions 200

9.1 Introduction 200

9.2 Data Types 200

9.2.1 tosca.datatypes.nfv.L2AddressData 200

9.2.1.1 Description 200

9.2.1.2 Properties 200

9.2.1.3 Definition 201

9.2.1.4 Examples 201

9.2.1.5 Additional Requirements 201

9.2.2 tosca.datatypes.nfv.L3AddressData 201

9.2.2.1 Description 201

9.2.2.2 Properties 201

9.2.2.3 Definition 202

9.2.2.4 Examples 202

9.2.2.5 Additional Requirements 203

9.2.3 tosca.datatypes.nfv.AddressData 203

9.2.3.1 Description 203

9.2.3.2 Properties 203

9.2.3.3 Definition 204

9.2.3.4 Examples 204

9.2.3.5 Additional Requirements 204

9.2.4 tosca.datatypes.nfv.ConnectivityType 204

9.2.4.1 Description 204

9.2.4.2 Properties 205

9.2.4.3 Definition 205

9.2.4.4 Examples 205

9.2.4.5 Additional Requirements 205

9.2.5 tosca.datatypes.nfv.LinkBitrateRequirements 206

9.2.5.1 Description 206

9.2.5.2 Properties 206

9.2.5.3 Definition 206

9.2.5.4 Examples 206

9.2.5.5 Additional Requirements 207

9.2.6 tosca.datatypes.nfv.CpProtocolData 207

9.2.6.1 Description 207

9.2.6.2 Properties 207

9.2.6.3 Definition 207

9.2.6.4 Examples 207

9.2.6.5 Additional Requirements 208

9.2.7 tosca.datatypes.nfv.Qos 208

9.2.7.1 Description 208

9.2.7.2 Properties 208

9.2.7.3 Definition 208

9.2.7.4 Examples 209

9.2.7.5 Additional Requirements 209

9.2.8 tosca.datatypes.nfv.VnfProfile 209

9.2.8.1 Description 209

9.2.8.2 Properties 209

9.2.8.3 Definition 209

9.2.8.4 Example 210

9.2.8.5 Additional Requirements 210

9.2.9 tosca.datatypes.nfv.VnfMonitoringParameter 210

9.2.9.1 Description 210

9.2.9.2 Properties 210

9.2.9.3 Definition 211

9.2.9.4 Examples 212

9.2.9.5 Additional Requirements 212

9.3 Artifact Types 212

9.4 Capabilities Types 212

9.4.1 tosca.capabilities.nfv.VirtualLinkable 212

9.4.1.1 Description 212

9.4.1.2 Properties 212

9.4.1.3 Definition 212

9.4.2 Void 212

9.5 Requirements Types 212

9.6 Relationship Types 213

9.6.1 tosca.relationships.nfv.VirtualLinksTo 213

9.6.1.1 Description 213

9.6.1.2 Properties 213

9.6.1.3 Definition 213

9.6.2 Void 213

9.6.3 tosca.relationships.nfv.VipVirtualLinksTo 213

9.6.3.1 Description 213

9.6.3.2 Properties 213

9.6.3.3 Definition 214

9.7 Interface Types 214

9.8 Node Types 214

9.8.1 tosca.nodes.nfv.Cp 214

9.8.1.1 Description 214

9.8.1.2 Properties 214

9.8.1.3 Attributes 215

9.8.1.4 Requirements 215

9.8.1.5 Capabilities 215

9.8.1.6 Definition 215

9.8.1.7 Additional requirements 216

9.9 Group Types 216

9.10 Policy Types 216

9.10.1 tosca.policies.nfv.Abstract.SecurityGroupRule 216

9.10.1.1 Description 216

9.10.1.2 Properties 216

9.10.1.3 Definition 217

9.10.1.4 Additional Requirements 218

Annex A (informative): Examples 219

A.1 Deployment flavour design mapping 219

A.1.1 Introduction 219

A.1.2 Design principle for VNFD deployment flavour 219

A.1.3 Design principle for NSD deployment flavour 220

A.2 VNFD with deployment flavour modelling design example 220

A.3 VNF external connection point 230

A.3.1 General 230

A.3.2 External connection point re-exposing an internal connection point 230

A.3.3 External connection point connected to an internal virtual link 232

A.4 VNFD modelling design example by using TOSCA composition 233

A.5 VNFD with Single deployment flavour modelling design example 236

A.6 Scaling and Instantiation Level examples 240

A.6.1 ScalingAspect and InstantiationLevels policies with uniform delta 240

A.6.2 ScalingAspect and InstantationLevels policies with non‑uniform deltas 245

A.7 Service Access Point 248

A.7.1 General 248

A.7.2 VNF External connection point exposing as a SAP 249

A.7.3 SAP connected to an NS virtual link 250

A.8 NSD with Single deployment flavour modelling design example 251

A.9 Mapping between NFV IM and TOSCA concepts 255

A.9.1 Introduction 255

A.9.2 Mapping between ETSI GS NFV-IFA 011 IM and TOSCA concepts 255

A.9.3 Mapping between ETSI GS NFV-IFA 014 IM and TOSCA concepts 256

A.10 PNFD modelling design example 257

A.11 NSD with Multiple deployment flavour modelling design example 258

A.12 NSD with nested NS design example 264

A.13 Virtual IP address connection point 269

A.14 NSD with VNF Forwarding Graph design example 274

Annex B (normative): etsi\_nfv\_sol001\_type definitions 282

B.1 Purpose 282

B.2 VNFD type definitions file 282

B.3 NSD type definitions file 282

B.4 PNFD type definitions file 283

B.5 Common type definitions file 283

Annex C (normative): Conformance 284

C.1 Purpose 284

C.2 NFV TOSCA YAML service template 284

C.3 NFV TOSCA processor 285

Annex D (informative): Mapping between properties of TOSCA types and API attributes 286

D.1 Introduction 286

D.2 VNFD-related constructs 286

D.3 NSD-related constructs 294

Annex E (informative): TOSCA Imperative workflows 299

E.1 Purpose 299

E.2 TOSCA Imperative workflows for the NSD 299

E.2.1 Introduction 299

E.2.2 Definition of an NS workflow 299

E.2.3 Examples 300

Annex F (informative): Non-Backward Compatible Changes in the present document 305

F.1 Introduction 305

F.2 Non-Backward Compatible changes between version 2.7.1 and 2.6.1 305

Annex G (informative): Change History 306

History 309

# 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](https://portal.etsi.org/Services/editHelp!/Howtostart/ETSIDraftingRules.aspx) (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".

[14] Hash Function Textual Names registry at IANA.

NOTE: Available at <https://www.iana.org/assignments/hash-function-text-names>.

[15] The Open Group Base Specifications Issue 7, 2018 edition IEEE™ Std 1003.1-2017 (Revision of IEEE™ Std 1003.1-2008).

NOTE: Available at <http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html>.

[16] IEEE™ 802.1Q-2014: "IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks".

[17] IETF RFC 791: "Internet Protocol".

NOTE: Available at <https://tools.ietf.org/html/rfc791>.

[18] IETF RFC 8200: "Internet Protocol, Version 6 (IPv6) Specification".

NOTE: Available at <https://tools.ietf.org/html/rfc8200>.

[19] IANA: "Assigned Internet Protocol Numbers".

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

[20] TOSCA-Simple-Profile-yaml-v1.3: "TOSCA Simple Profile in YAML Version 1.3".

[21] IETF RFC 6225: "Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information".

NOTE: Available at <https://tools.ietf.org/html/rfc6225>.

## 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] Void.

[i.6] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package and PNFD Archive 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".

[i.9] ETSI GS NFV-SOL 003: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

[i.10] ETSI GS NFV-SOL 005: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

[i.11] ETSI GS NFV-SOL 007: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Network Service Descriptor File Structure Specification".

[i.12] OpenStack® documentation: "Disk and container formats for images".

NOTE 1: Available at <https://docs.openstack.org/glance/pike/user/formats.html>.

NOTE 2: The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. ETSI is not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

[i.13] Openstack® Metadata service.

NOTE: Available at <https://docs.openstack.org/nova/rocky/user/metadata-service.html>.

[i.14] Openstack® User-data service.

NOTE: Available at <https://docs.openstack.org/nova/rocky/user/user-data.html>.

[i.15] Openstack® Personality service.

NOTE: Available at <http://www.openstacknetsdk.org/docs/html/T_net_openstack_Core_Domain_Personality.htm>.

[i.16] ETSI NFV registry of VimConnectionInfo information.

NOTE: Available at <https://nfvwiki.etsi.org/index.php?title=NFV_Registries>.

# 3 Definition of terms, symbols 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 Symbols

Void.

## 3.3 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].

Three main deployment templates are identified in the present document:

* The Virtualised Network Function Descriptor (VNFD).
* The Network Service Descriptor (NSD).
* The Physical Network Function Descriptor (PNFD).

## 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 as defined in ETSI GS NFV-IFA 014 [2]. The NSD:

* References zero, one or more Virtualised Network Function Descriptors (VNFD).
* References zero, one or more Physical Network Functions Descriptors (PNFD).
* References zero, one or more nested NSD.
* Includes zero, one or more Virtual Link Descriptors (VLD).
* Includes 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.

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). AVDU 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.

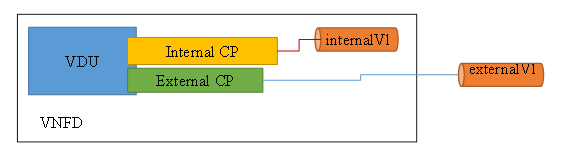


Figure 5.3-1: Overview of VNF descriptor

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 Physical Network Function Descriptor

The Physical Network Function Descriptor (PNFD) information element is a deployment template enabling on-boarding PNFs and referencing them from an NSD. It focuses on connectivity aspects only.

## 5.5 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], unless VNF indicators are used, in which case the "tosca\_definitions\_version" keyword shall point to TOSCA-Simple-Profile-YAML-v1.3 [20].

NOTE: In the present version of the present document, only the notification feature defined in TOSCA-Simple-Profile-YAML-v1.3 [20] is used.

Table 5.5-1 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 5.5-1

|  |  |
| --- | --- |
| Namespace Prefix | Specification Description |
| toscanfv | The TOSCA namespace prefix of all the TOSCA types as specified in the present document. |

## 5.6 Imports statement

### 5.6.1 VNFD TOSCA service template

A VNFD TOSCA service template as specified in clause 6.11 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.
* Others, as described in clause 6.11.

As specified in TOSCA-Simple-Profile-yaml-v1.2 [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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_vnfd\_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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_vnfd\_types.yaml  - file: any\_other\_files.yaml  - file: custom\_vnfd\_datatypes\_extension.yaml |

### 5.6.2 NSD TOSCA service template

An NSD TOSCA service template as specified in clause 7.11 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.
* Others, as described in clause 7.11.

As specified in TOSCA-Simple-Profile-yaml-v1.2 [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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_nsd\_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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_nsd\_types.yaml  - file: any\_other\_files.yaml  - file: custom\_nsd\_node\_types\_extension.yaml |

### 5.6.3 PNFD TOSCA service template

A PNFD TOSCA service template as specified in clause 8.11 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.
* Others, as described in clause 8.11.

As specified in TOSCA-Simple-Profile-yaml-v1.2 [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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_pnfd\_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://forge.etsi.org/rep/nfv/SOL001/raw/v2.6.1/etsi\_nfv\_sol001\_pnfd\_types.yaml  - file: any\_other\_files.yaml  - file: custom\_pnfd\_node\_types\_extension.yaml |

## 5.7 Type extension

### 5.7.1 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.

### 5.7.2 Rules

Type extension may be applied to NFV types defined in the present document within the limits specified table 5.7.3-1, 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 and attributes may be introduced with no restriction within the limits specified table 5.7.3-1.

Existing properties may be extended according to the following rules:

1. 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).
2. 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.
3. 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).
4. 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.

### 5.7.3 VNFD Types

Table 5.7.3-1 specifies the extension point where VNFD author may extend the pre-defined types.

Table 5.7.3-1: 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)  See note 4. |
| interfaces | Vnflcm.{operation\_name}.inputs.additional\_parameters (as a new property)  Vnflcm.{operation\_name}.inputs (as new properties). See note 4. |
| attributes | One attribute of primitive type per VNF indicator may be added. |
| tosca.nodes.nfv.Vdu.Compute | properties | configurable\_properties (as a new property). See note 2 and note 4. |
| tosca.datatypes.nfv.VnfInfoModifiableAttributes | properties | extensions (as a new property)  metadata (as a new property). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfConfigurableProperties | properties | additional\_configurable\_properties (as a new property). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfcConfigurableProperties | properties | additional\_vnfc\_configurable\_properties (as a new property). See note 4. |
| tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions | properties | (new properties). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata | properties | (new properties). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfAdditionalConfigurableProperties | properties | (new properties). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfAdditionalVnfcConfigurableProperties | properties | (new properties). See note 1 and note 4. |
| tosca.datatypes.nfv.VnfOperationAdditionalParameters | properties | (new properties). See note 1 and note 4. |
| tosca.interfaces.nfv.VnfIndicator | notifications | one notification may be added per Vnf indicator. See note 3. |
| NOTE 1: VNF specific extension datatypes should be given names starting by the provider name followed by a dot (".") in order to avoid collisions when importing these datatypes in an NSD service template. The provider name may, but need not, be identical to the value of the provider property of the VNF node type. Furthermore, it is the VNF provider’s responsibility to ensure the uniqueness of the names of its datatypes, i.e. the datatype names starting with its provider name.  NOTE 2: VNF specific Vdu.Compute node types should be given names starting by the provider name followed by a dot (".") in order to avoid collisions if these node types are imported in an NSD service template. The provider name may, but need not, be identical to the value of the provider property of the VNF node type. Furthermore, it is the VNF provider’s responsibility to ensure the uniqueness of the names of its Vdu.Compute node types, i.e. the Vdu.Compute node type names starting with its provider name.  NOTE 3: VNF specific interface types should be given names starting by the provider name followed by a dot (".") in order to avoid collisions when importing these types in an NSD service template. The provider name may, but need not, be identical to the value of the provider property of the VNF node type. Furthermore, it is the VNF provider’s responsibility to ensure the uniqueness of the names of its interface types, i.e. the interface type names starting with its provider name.  NOTE 4: If a property is defined with a required value equal to false, the default value shall not be present in VNFD. This also applies to any new datatypes introduced in the VNFD. | | |

### 5.7.4 NSD types

Table 5.7.4-1 specifies the extension points where NSD author may extend the pre-defined types.

Table 5.7.4-1: NSD type extension points

|  |  |  |
| --- | --- | --- |
| **Type** | **Keyname** | **Property name** |
| tosca.datatypes.nfv.NsOperationAdditionalParameters | properties | (new properties). See note. |
| NOTE: If a property is defined with a required value equal to false, the default value shall not be present in NSD. This also applies to any new datatypes introduced in the NSD. | | |

## 5.8 Non-Backward Compatible changes

Annex F provides the list of non-backward compatible changes during the development of the present document.

# 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: The autoScale feature specified in ETSI GS NFV-IFA 011 [1] is not supported in this version of the present document.

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 tosca.nodes.nfv.VduCp, tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.Vdu.VirtualBlockStorage, tosca.nodes.nfv.Vdu.VirtualObjectStorage and tosca.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.

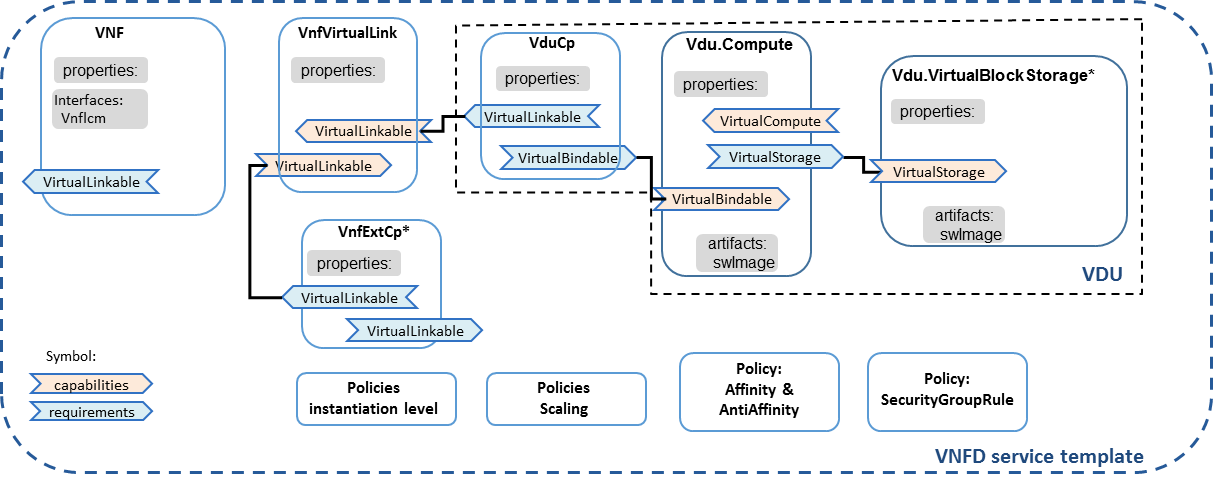


Figure 6.1-1: Service template VNFD overview

## 6.2 Data Types

### 6.2.1 tosca.datatypes.nfv.CpProtocolData

#### 6.2.1.1 Description

The CpProtocolData data type is defined in clause 9.2.6 of the present document.

### 6.2.2 tosca.datatypes.nfv.AddressData

#### 6.2.2.1 Description

The AddressData data type is defined in clause 9.2.3 of the present document.

### 6.2.3 tosca.datatypes.nfv.L2AddressData

#### 6.2.3.1 Description

The L2AddressData data type is defined in clause 9.2.1 of the present document.

### 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 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: tosca.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: tosca.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 tosca.datatypes.nfv.L3AddressData

#### 6.2.5.1 Description

The L3AddressData data type is defined in clause 9.2.2 of the present document.

### 6.2.6 tosca.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: tosca.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 type supports 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 | yes | 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: true  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](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: tosca.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: tosca.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 tosca.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 virtual\_cpu\_pinning\_rule. |
| 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: tosca.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** | VnfcConfigurableProperties |
| **Type Qualified Name** | toscanfv:VnfcConfigurableProperties |
| **Type URI** | tosca.datatypes.nfv.VnfcConfigurableProperties |

#### 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 |
| --- | --- | --- | --- | --- |
| additional\_vnfc\_configurable\_properties | no | tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties |  | Describes additional configuration for VNFC that can be modified using the ModifyVnfInfo 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** | VnfcAdditionalConfigurableProperties |
| **Type Qualified Name** | toscanfv:VnfcAdditionalConfigurableProperties |
| **Type URI** | tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties |

#### 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: tosca.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** | VduProfile |
| **Type Qualified Name** | toscanfv:VduProfile |
| **Type URI** | tosca.datatypes.nfv.VduProfile |

#### 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: tosca.nodes.nfv.Vdu.VirtualBlockStorage  properties:  # omitted for brevity  capabilities:  virtual\_storage  VirtualStorage\_A2:  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage  properties:  # omitted for brevity  capabilities:  virtual\_storage  VduCp\_A1:  type: tosca.nodes.nfv.VduCp  properties:  # omitted for brevity  requirements:  - virtual\_binding: VDU\_A  - virtual\_link  VduCp\_A2:  type: tosca.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.VlProfile

#### 6.2.13.1 Description

The VlProfile 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** | VlProfile |
| **Type Qualified Name** | toscanfv:VlProfile |
| **Type URI** | tosca.datatypes.nfv.VlProfile |

#### 6.2.13.2 Properties

The properties of the VlProfile 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 VlProfile data type shall comply with the following definition:

|  |
| --- |
| tosca.datatypes.nfv.VlProfile:  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: tosca.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: tosca.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 tosca.datatypes.nfv.L2ProtocolData

#### 6.2.15.1 Description

The L2ProtocolData data type escribes 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 | yes | 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. |
| segmentation\_id | no | string |  | If present, specifies a specific virtualised network segment, which depends on the network type. For e.g. VLAN ID for VLAN network type and tunnel ID for GRE/VXLAN network types.  See note. |
| NOTE: If this property is included in the VNFD, the property value shall be provided at run-time, unless a default value is provided at design time in the VNFD. If a default value is provided at design-time, this value may be overridden at run-time. | | | | |

#### 6.2.15.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.L2ProtocolData:  derived\_from: tosca.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: true  default: false  mtu:  type: integer  description: Specifies the maximum transmission unit (MTU) value for this L2 protocol.  required: false  constraints:  - greater\_than: 0  segmentation\_id:  type: string  description: Specifies a specific virtualised network segment, which depends on the network type. For e.g., VLAN ID for VLAN network type and tunnel ID for GRE/VXLAN network types  required: false |

#### 6.2.15.4 Examples

See example in clause A.5.

#### 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: tosca.datatypes.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 tosca.datatypes.nfv.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: tosca.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: tosca.datatypes.nfv.ScaleInfo |

#### 6.2.18.4 Examples

See clause A.6.

#### 6.2.18.5 Additional Requirements

None.

### 6.2.19 tosca.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: tosca.datatypes.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 tosca.datatypes.nfv.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: tosca.datatypes.Root  description: Represents information to configure lifecycle management operations  properties:  instantiate:  type: tosca.datatypes.nfv.VnfInstantiateOperationConfiguration  description: Configuration parameters for the InstantiateVnf operation  required: false  scale:  type: tosca.datatypes.nfv.VnfScaleOperationConfiguration  description: Configuration parameters for the ScaleVnf operation  required: false  scale\_to\_level:  type: tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration  description: Configuration parameters for the ScaleVnfToLevel operation  required: false  change\_flavour:  type: tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration  description: Configuration parameters for the changeVnfFlavourOpConfig operation  required: false  heal:  type: tosca.datatypes.nfv.VnfHealOperationConfiguration  description: Configuration parameters for the HealVnf operation  required: false  terminate:  type: tosca.datatypes.nfv.VnfTerminateOperationConfiguration  description: Configuration parameters for the TerminateVnf operation  required: false  operate:  type: tosca.datatypes.nfv.VnfOperateOperationConfiguration  description: Configuration parameters for the OperateVnf operation  required: false  change\_ext\_connectivity:  type: tosca.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 tosca.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]. This data type definition is reserved for future use in the present document. 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: tosca.datatypes.Root  description: represents information that affect the invocation of the InstantiateVnf operation.  # This data type definition is reserved for future use in the present document.  # properties: |

#### 6.2.21.4 Examples

None.

#### 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** | VnfScaleOperationConfiguration |
| **Type Qualified Name** | toscanfv:VnfScaleOperationConfiguration |
| **Type URI** | tosca.datatypes.nfv.VnfScaleOperationConfiguration |

#### 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 |
| --- | --- | --- | --- | --- |
| scaling\_by\_more\_than\_one\_step\_supported | yes | boolean |  | Signals whether passing a value larger than one in the numScalingSteps parameter of the ScaleVnf operation is supported by this VNF.  Default is FALSE, 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: tosca.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: true  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** | VnfScaleToLevelOperationConfiguration |
| **Type Qualified Name** | toscanfv:VnfScaleToLevelOperationConfiguration |
| **Type URI** | tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration |

#### 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 |
| --- | --- | --- | --- | --- |
| arbitrary\_target\_levels\_supported | 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: tosca.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: tosca.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 tosca.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: tosca.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** | ScaleInfo |
| **Type Qualified Name** | toscanfv:ScaleInfo |
| **Type URI** | tosca.datatypes.nfv.ScaleInfo |

#### 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 |
| --- | --- | --- | --- | --- |
| scale\_level | yes | integer | greater\_or\_equal  : 0 | 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: tosca.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 with regards to 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: tosca.datatypes.Root  description: describes the details of an aspect used for horizontal scaling  properties:  name:  type: string  description: Human readable name of the aspect  required: true  description:  type: string  description: Human readable description of the aspect  required: true  max\_scale\_level:  type: integer # positiveInteger  description: 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  required: true  constraints:  - greater\_or\_equal: 0  step\_deltas:  type: list  description: 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)  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 is defined in clause 9.2.5 of the present document.

### 6.2.30 tosca.datatypes.nfv.ConnectivityType

#### 6.2.30.1 Description

The ConnectivityType data type is defined in clause 9.2.4 of the present document.

### 6.2.31 tosca.datatypes.nfv.VnfConfigurableProperties

#### 6.2.31.1 Description

The VnfConfigurableProperties data type describes configurable properties for a given VNF. Configurable properties can be standardized as listed below (e.g. related to auto scaling, auto healing and interface configuration) or can be VNF-specific as defined by the VNF provider.

The value of all VNF configurable properties listed in table 6.2.31.2-1 shall be modifiable anytime (including after instantiation of the VNF) via the Modify VNF information operation, unless stated otherwise in the description of the specific VNF configurable property.

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** | VnfConfigurableProperties |
| **Type Qualified Name** | toscanfv:VnfConfigurableProperties |
| **Type URI** | tosca.datatypes.nfv.VnfConfigurableProperties |

#### 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 |
| --- | --- | --- | --- | --- |
| is\_autoscale\_enabled | no | boolean |  | It permits to enable (TRUE)/disable (FALSE) the auto-scaling functionality. If the property is not present, then configuring this VNF property is not supported. |
| is\_autoheal\_enabled | no | boolean |  | It permits to enable (TRUE)/disable (FALSE) the auto-healing functionality. If the property is not present, then configuring this VNF property is not supported. |
| vnfm\_interface\_info | no | tosca.datatypes.nfv.VnfmInterfaceInfo |  | Contains information enabling the VNF instance to access to the NFV-MANO interfaces produced by the VNFM (e.g. URIs and credentials).  If the property is not present, then configuring this VNF property is not supported.  If this attribute is declared for a VNF, its initial value shall be set prior to or at instantiation time (as initial value in the VNFD or via the VNF LCM interface). Its value shall be further modifiable after instantiation via the Modify VNF information operation. |
| vnfm\_oauth\_server\_info | no | tosca.datatypes.nfv.OauthServerInfo |  | Contains information to enable discovery of the authorization server protecting access to VNFM interfaces.  If the property is not present, then configuring this VNF property is not supported.  If this attribute is declared for a VNF, its initial value shall be set prior to or at instantiation time (as initial value in the VNFD or via the VNF LCM interface). Its value shall be further modifiable after instantiation via the Modify VNF information operation. |
| vnf\_oauth\_server\_info | no | tosca.datatypes.nfv.OauthServerInfo |  | Contains information to enable discovery of the authorization server to validate the access tokens provided by the VNFM when the VNFM accesses the VNF interfaces, if that functionality (token introspection) is supported by the authorization server.  If the property is not present, then configuring this VNF property is not supported.  If this attribute is declared for a VNF, its initial value shall be set prior to or at instantiation time (as initial value in the VNFD or via the VNF LCM interface). Its value shall be further modifiable after instantiation via the Modify VNF information operation. |
| additional\_configurable\_property | no | tosca.datatypes.nfv.VnfAdditionalConfigurableProperties |  | It provides VNF specific configurable properties that can be modified using the ModifyVnfInfo operation.  If some of these properties are declared as required, their values shall be set prior to or at instantiation time (as initial value in the VNFD or via the VNF LCM interface). Their values may be modifiable after instantiation via the Modify VNF information operation if such modification of individual attributes is supported by the VNF and declared per attribute in the VNFD. |

#### 6.2.31.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfConfigurableProperties:  derived\_from: tosca.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 property is not present, then configuring this 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 property is not present, then configuring this VNF property is not supported.  required: false  vnfm\_interface\_info:  type: tosca.datatypes.nfv.VnfmInterfaceInfo  description: Contains information enabling access to the NFV-MANO interfaces produced by the VNFM (e.g. URIs and credentials). If the property is not present, then configuring this VNF property is not supported.  required: false  vnfm\_oauth\_server\_info:  type: tosca.datatypes.nfv.OauthServerInfo  description: Contains information to enable discovery of the authorization server protecting access to VNFM interfaces. If the property is not present, then configuring this VNF property is not supported.  required: false  vnf\_oauth\_server\_info:  type: tosca.datatypes.nfv.OauthServerInfo  description: Contains information to enable discovery of the authorization server to validate the access tokens provided by the VNFM when the VNFM accesses the VNF interfaces, if that functionality (token introspection) is supported by the authorization server. If the property is not present, then configuring this 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: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties  # derived types are expected to introduce  # additional\_configurable\_properties with its type derived from  # tosca.datatypes.nfv.VnfAdditionalConfigurableProperties |

#### 6.2.31.4 Examples

Example definition of configurable properties without properties assignment value.

|  |
| --- |
| tosca\_definitions\_version: tosca\_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 ]  configurable\_properties:  type: MyCompany.datatypes.nfv.VnfConfigurableProperties  data\_types:  Mycompany.datatypes.nfv.VnfConfigurableProperties:  derived\_from: tosca.datatypes.nfv.VnfConfigurableProperties  properties:  additional\_configurable\_properties:  type: MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties  MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties:  derived\_from: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties  properties:  name\_prefix\_in\_vim:  type: string  required: false  dns\_server:  type: string  required: true |

In the above example, properties definitions are provided and properties assignment values are not necessary. The properties values are available in the API.

Example definition of configurable properties with properties assignment value.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

...

node\_types:

MyCompany.SunshineDB.1\_0.1\_0:

derived\_from: tosca.nodes.nfv.VNF

interfaces:

Vnflcm:

type: tosca.interfaces.nfv.Vnflcm

MyCompany.nodes.nfv.Vdu.Aux:

derived\_from: tosca.nodes.nfv.Vdu.Compute

properties:

configurable\_properties:

type: MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties

required: false

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: true

default: "MyCustomer"

dns\_server:

type: string

required: true

default: "90.200.250.57"

topology\_template:

substitution\_mappings:

node\_type: MyCompany.SunshineDB.1\_0.1\_0

requirements:

virtual\_link: [ dbBackendIpv4, virtual\_link ] # IPv4 for SQL

inputs:

name\_prefix\_in\_vim:

type: string

dns\_server:

type: string

node\_templates:

SunshineDB:

type: MyCompany.SunshineDB.1\_0.1\_0

dbBackend:

type: MyCompany.nodes.nfv.Vdu.Aux

properties:

...

configurable\_properties:

additional\_vnfc\_configurable\_properties:

name\_prefix\_in\_vim: { get\_input: name\_prefix\_in\_vim }

dns\_server: { get\_input: dns\_server }

In the above example, default values are provided in the node type definition, properties assignment by using TOSCA get\_input function is described in the node template. The properties values from the API will override the default values.

#### 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** | VnfAdditionalConfigurableProperties |
| **Type Qualified Name** | toscanfv:VnfAdditionalConfigurableProperties |
| **Type URI** | tosca.datatypes.nfv.VnfAdditionalConfigurableProperties |

#### 6.2.32.2 Properties

The properties of the VnfAdditionalConfigurableProperties data type shall comply with the provisions set out in table 6.2.32.2-1.

Table 6.2.32.2-1: Properties

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| is\_writable\_anytime | yes | boolean |  | It specifies whether these additional configurable properties are writeable (TRUE) at any time (i.e. prior to / at instantiation time as well as after instantiation).or (FALSE) only prior to / at instantiation time.  If this property is not present, the additional configurable properties are writable anytime. |

#### 6.2.32.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfAdditionalConfigurableProperties:  derived\_from: tosca.datatypes.Root  description: is an empty base type for deriving data types for describing additional configurable properties for a given VNF  properties:  is\_writable\_anytime:  type: boolean  description: It specifies whether these additional configurable properties are writeable (TRUE) at any time (i.e. prior to / at instantiation time as well as after instantiation).or (FALSE) only prior to / at instantiation time. If this property is not present, the additional configurable properties are writable anytime.  required: true  default: true |

#### 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** | VnfInfoModifiableAttributes |
| **Type Qualified Name** | toscanfv:VnfInfoModifiableAttributes |
| **Type URI** | tosca.datatypes.nfv.VnfInfoModifiableAttributes |

#### 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: tosca.datatypes.Root  description: Describes VNF-specific extension and metadata for a given VNF  #properties:  #extensions:  #type: tosca.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  # tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions  #metadata:  #type: tosca.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  # tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata |

#### 6.2.33.4 Examples

See clause 6.8.1.9.

#### 6.2.33.5 Additional Requirements

None.

### 6.2.34 tosca.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: tosca.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 tosca.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: tosca.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: tosca\_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.Qos

#### 6.2.36.1 Description

The Qos data type is defined in clause 9.2.7 of the present document.

### 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 |  | 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.  More information regarding the usage of this property is available at:  <https://register.etsi.org> |

#### 6.2.37.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.LogicalNodeData:  derived\_from: tosca.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 tosca.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. |
| provider | no | string |  | Provider of this software image. |
| checksum | yes | tosca.datatypes.nfv.ChecksumData |  | 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.  See note 1. |
| 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.  See note 2. |
| min\_disk | yes | scalar-unit.size | greater\_or\_equal: 0 B | The minimal disk size requirement for this software image. |
| min\_ram | no | scalar-unit.size | greater\_or\_equal: 0 B | 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. |
| NOTE 1: The list of permitted values was taken from "Container formats" in [i.12].  NOTE 2: The list of permitted values was adapted from "Disk formats" in [i.12]. | | | | |

#### 6.2.38.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.SwImageData:  derived\_from: tosca.datatypes.Root  description: describes information related to a software image artifact  properties:  name:  type: string  description: Name of this software image  required: true  version:  type: string  description: Version of this software image  required: true  provider:  type: string  description: Provider of this software image  required: false  checksum:  type: tosca.datatypes.nfv.ChecksumData  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  constraints:  - greater\_or\_equal: 0 B  min\_ram:  type: scalar-unit.size # Number  description: The minimal RAM requirement for this software image  required: false  constraints:  - greater\_or\_equal: 0 B  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** | VirtualBlockStorageData |
| **Type Qualified Name** | toscanfv:VirtualBlockStorageData |
| **Type URI** | tosca.datatypes.nfv.VirtualBlockStorageData |

#### 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 |
| --- | --- | --- | --- | --- |
| size\_of\_storage | yes | scalar-unit.size | greater\_or\_equal:0 B | Size of virtualised storage resource |
| vdu\_storage\_requirements | 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> |
| rdma\_enabled | yes | 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: tosca.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  constraints:  - greater\_or\_equal: 0 B  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: true  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** | VirtualObjectStorageData |
| **Type Qualified Name** | toscanfv:VirtualObjectStorageData |
| **Type URI** | tosca.datatypes.nfv.VirtualObjectStorageData |

#### 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 |
| --- | --- | --- | --- | --- |
| max\_size\_of\_storage | no | scalar-unit.size | greater\_or\_equal:0 B | Maximum size of virtualised 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: tosca.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 virtualised storage resource  required: false  constraints:  - greater\_or\_equal: 0 B |

#### 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** | VirtualObjectFileData |
| **Type Qualified Name** | toscanfv:VirtualFileStorageData |
| **Type URI** | tosca.datatypes.nfv.VirtualFileStorageData |

#### 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 |
| --- | --- | --- | --- | --- |
| size\_of\_storage | yes | scalar-unit.size | greater\_or\_equal:0 B | Size of virtualised storage resource |
| file\_system\_protocol | yes | string |  | 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: tosca.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 virtualised storage resource  required: true  constraints:  - greater\_or\_equal: 0 B  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** | VirtualLinkBitrateLevel |
| **Type Qualified Name** | toscanfv:VirtualLinkBitrateLevel |
| **Type URI** | tosca.datatypes.nfv.VirtualLinkBitrateLevel |

#### 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 |
| --- | --- | --- | --- | --- |
| bitrate\_requirements | yes | tosca.datatypes.nfv.LinkBitrateRequirements |  | 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: tosca.datatypes.Root  description: Describes bitrate requirements applicable to the virtual link instantiated from a particular VnfVirtualLink  properties:  bitrate\_requirements:  type: tosca.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: true  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]. This data type definition is reserved for future use in the present document. 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** | VnfChangeFlavourOperationConfiguration |
| **Type Qualified Name** | toscanfv:VnfChangeFlavourOperationConfiguration |
| **Type URI** | tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration |

#### 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: tosca.datatypes.Root  description: represents information that affect the invocation of the ChangeVnfFlavour operation  # This data type definition is reserved for future use in the present document.  # properties: |

#### 6.2.44.4 Examples

None.

#### 6.2.44.5 Additional Requirements

None.

### 6.2.45 tosca.datatypes.nfv.VnfChangeExtConnectivityOperation Configuration

#### 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]. This data type definition is reserved for future use in the present document. 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** | VnfChangeExtConnectivityOperationConfiguration |
| **Type Qualified Name** | toscanfv:VnfChangeExtConnectivityOperationConfiguration |
| **Type URI** | tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration |

#### 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: tosca.datatypes.Root  description: represents information that affect the invocation of the ChangeExtVnfConnectivity operation  # This data type definition is reserved for future use in the present document.  # properties: |

#### 6.2.45.4 Examples

None.

#### 6.2.45.5 Additional Requirements

None.

### 6.2.46 tosca.datatypes.nfv.VnfMonitoringParameter

The VnfMonitoringParameter data type is defined in clause 9.2.9 of the present document.

### 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** | VnfcMonitoringParameter |
| **Type Qualified Name** | toscanfv:VnfcMonitoringParameter |
| **Type URI** | tosca.datatypes.nfv.VnfcMonitoringParameter |

#### 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  v\_cpu\_usage\_mean  v\_cpu\_usage\_peak  v\_memory\_usage\_mean  v\_memory\_usage\_peak  v\_disk\_usage\_mean  v\_disk\_usage\_peak  v\_net\_byte\_incoming  v\_net\_byte\_outgoing  v\_net\_packet\_incoming  v\_net\_packet\_outgoing  usage\_mean\_vStorage  usage\_peak\_vStorage | Identifies a performance metric to be monitored.  Performance metric values shall be either set to:   * A corresponding measurement name defined in clause 7.2 of ETSI GS NFV‑IFA 027 [7], without appending a sub‑counter. In this case the VNFM computes these measurements from lower-level metrics collected from the VIM. See note. * A corresponding measurement name defined in clause 7.1 of ETSI GS NFV‑IFA 027 [7], without appending a sub‑counter. In this case the VNFM collects these metrics from the VIM for all compute, storage and network resources allocated to the VNFC instance. |
| collection\_period | no | scalar-unit.time |  | Describes the periodicity at which to collect the performance information. |
| NOTE: The measured object type for \_cpu\_usage\_mean\_vnf, v\_cpu\_usage\_peak\_vnf, v\_memory\_usage\_mean\_vnf, v\_memory\_usage\_peak\_vnf, v\_disk\_usage\_mean\_vnf and v\_disk\_usage\_peak\_vnf is the VNFC. | | | | |

#### 6.2.47.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfcMonitoringParameter:  derived\_from: tosca.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 to be monitored, 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, v\_cpu\_usage\_mean, v\_cpu\_usage\_peak,v\_memory\_usage\_mean,v\_memory\_usage\_peak, v\_disk\_usage\_mean, v\_disk\_usage\_peak, v\_net\_byte\_incoming, v\_net\_byte\_outgoing, v\_net\_packet\_incoming, v\_net\_packet\_outgoing, usage\_mean\_vStorage, usage\_peak\_vStorage ]  collection\_period:  type: scalar-unit.time  description: Describes the 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** | VirtualLinkMonitoringParameter |
| **Type Qualified Name** | toscanfv: VirtualLinkMonitoringParameter |
| **Type URI** | tosca.datatypes.nfv.VirtualLinkMonitoringParameter |

#### 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** |
| --- | --- | --- | --- | --- |
| name | no | string |  | Human readable name of the monitoring parameter. |
| performance\_metric | yes | string | valid values:  byte\_incoming  byte\_outgoing  packet\_incoming  packet\_outgoing | Identifies a performance metric to be monitored.  Performance metric values shall be set to following measurement names defined in clause 7.1 of ETSI GS NFV-IFA 027 [7], without appending a sub-counter:   * ByteIncoming * ByteOutgoing * PacketIncoming * PacketOutgoing   The VNFM collects these metrics from the VIM by aggregating the sub-counters of all virtual link ports attached to the virtual link to which the metrics apply. |
| collection\_period | no | scalar-unit.time |  | Describes the periodicity at which to collect the performance information. |

#### 6.2.48.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VirtualLinkMonitoringParameter:  derived\_from: tosca.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 to be monitored.  required: true  constraints:  - valid\_values: [ byte\_incoming, byte\_outgoing, packet\_incoming, packet\_outgoing ]  collection\_period:  type: scalar-unit.time  description: Describes the 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 tosca.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: tosca.datatypes.Root  description: information used to access an interface exposed by a VNF  properties:  uri\_components:  type: tosca.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 tosca.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: tosca.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: tosca.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 tosca.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: tosca.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 if out of service or is not reachable, the VNFM is expected to try reaching the interface through another CP instance.

### 6.2.52 tosca.datatypes.nfv.VnfProfile

#### 6.2.52.1 Description

The VnfProfile data type is defined in clause 9.2.8 of the present document.

### 6.2.53 tosca.datatypes.nfv.ChecksumData

#### 6.2.53.1 Description

The ChecksumData data type describes information about the result of performing a checksum operation over some arbitrary data. Table 6.2.53.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.53.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | ChecksumData |
| **Type Qualified Name** | toscanfv:ChecksumData |
| **Type URI** | tosca.datatypes.nfv.ChecksumData |

#### 6.2.53.2 Properties

The properties of the ChecksumData data type shall comply with the provisions set out in table 6.2.53.2-1.

Table 6.2.53.2-1: Properties

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| algorithm | yes | string | Valid values: sha-224, sha-256, sha-384, sha-512 | Describes the algorithm used to obtain the checksum value, as described in [14]. |
| hash | yes | string |  | Contains the result of applying the algorithm indicated by the algorithm property to the data to which this ChecksumData refers. |

#### 6.2.53.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.ChecksumData:  derived\_from: tosca.datatypes.Root  description: Describes information about the result of performing a checksum operation over some arbitrary data  properties:  algorithm:  type: string  description: Describes the algorithm used to obtain the checksum value  required: true  constraints:  - valid\_values: [sha-224, sha-256, sha-384, sha-512 ]  hash:  type: string  description: Contains the result of applying the algorithm indicated by the algorithm property to the data to which this ChecksumData refers  required: true |

#### 6.2.53.4 Examples

|  |
| --- |
| <some\_tosca\_entity>:  properties:  checksum:  algorithm: sha-256  hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d |

#### 6.2.53.5 Additional Requirements

None.

### 6.2.54 tosca.datatypes.nfv.VnfmInterfaceInfo

#### 6.2.54.1 Description

The VnfmInterfaceInfo data type describes information enabling the VNF instance to access the NFV-MANO interfaces produced by the VNFM (e.g. URIs and credentials). Table 6.2.54.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.2 [3].

Table 6.2.54.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VnfmInterfaceInfo |
| **Type Qualified Name** | toscanfv:VnfmInterfaceInfo |
| **Type URI** | tosca.datatypes.nfv.VnfmInterfaceInfo |

#### 6.2.54.2 Properties

The properties of the VnfmInterfaceInfo data type shall comply with the provisions set out in table 6.2.54.2-1.

Table 6.2.54.2-1: Properties

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| interface\_name | yes | string | Valid values  vnf\_lcm  vnf\_pm  vnf\_fm | Identifies an interface produced by the VNFM. |
| details | no | tosca.datatypes.nfv.InterfaceDetails |  | Provide additional data to access the interface endpoint (e.g. API URI prefix). |
| credentials | no | map of string |  | Provides credential enabling access to the interface. This property is reserved for future use in the present document. |

#### 6.2.54.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfmInterfaceInfo:  derived\_from: tosca.datatypes.Root  description: Describes information enabling the VNF instance to access the NFV-MANO interfaces produced by the VNFM  properties:  interface\_name:  type: string  description: Identifies an interface produced by the VNFM.  required: true  constraints:  - valid\_values: [ vnf\_lcm, vnf\_pm, vnf\_fm ]  details:  type: tosca.datatypes.nfv.InterfaceDetails  description: Provide additional data to access the interface endpoint  required: false  credentials:  type: map  description: Provides credential enabling access to the interface  required: false  entry\_schema:  type: string |

#### 6.2.54.4 Examples

None.

#### 6.2.54.5 Additional Requirements

None.

### 6.2.55 tosca.datatypes.nfv.OauthServerInfo

#### 6.2.55.1 Description

The OauthServerInfo data type describes information to enable discovery of the authorization server. This data type definition is reserved for future use in the present document. Table 6.2.55.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.2 [3].

Table 6.2.55.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | OauthServerInfo |
| **Type Qualified Name** | toscanfv:OauthServerInfo |
| **Type URI** | tosca.datatypes.nfv.OauthServerInfo |

#### 6.2.55.2 Properties

None.

#### 6.2.55.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.OauthServerInfo:  derived\_from: tosca.datatypes.Root  description: information to enable discovery of the authorization server  #properties:  #This data type definition is reserved for future use in the present document |

#### 6.2.55.4 Examples

None.

#### 6.2.55.5 Additional Requirements

None.

### 6.2.56 tosca.datatypes.nfv.BootData

#### 6.2.56.1 Description

The BootData data type describes the information used to customize a virtualised compute resource at boot time. Table 6.2.56.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.2 [3].

Table 6.2.56.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | BootData |
| **Type Qualified Name** | toscanfv:BootData |
| **Type URI** | tosca.datatypes.nfv.BootData |

#### 6.2.56.2 Properties

The properties of the BootData data type shall comply with the provisions set out in table 6.2.56.2-1.

Table 6.2.56.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| vim\_specific\_properties | no | tosca.datatypes.nfv.BootDataVimSpecificProperties |  | Properties used for selecting VIM specific capabilities when setting the boot data. |
| kvp\_data | no | tosca.datatypes.nfv.KvpData |  | A set of key-value pairs for configuring a virtual compute resource. The mechanisms for conveying these key-value pairs to the virtual compute resource are out of the scope of the present document. An example of such mechanisms is the OpenStack metadata service defined in [i.13]. |
| content\_or\_file\_data | no | tosca.datatypes.nfv.ContentOrFileData |  | A string content or a file for configuring a virtual compute resource. The mechanisms for conveying the string content or the file to the virtual compute resource are out of the scope of the present document. An example of such mechanisms is the OpenStack User-data service defined in [i.14]. |

#### 6.2.56.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.BootData:  derived\_from: tosca.datatypes.Root  description: describes the information used to customize a virtualised compute resource at boot time.  properties:  vim\_specific\_properties:  type: tosca.datatypes.nfv.BootDataVimSpecificProperties  description: Properties used for selecting VIM specific capabilities when setting the boot data.  required: false  kvp\_data:  type: tosca.datatypes.nfv.KvpData  description: A set of key-value pairs for configuring a virtual compute resource.  required: false  content\_or\_file\_data:  type: tosca.datatypes.nfv.ContentOrFileData  description: A string content or a file for configuring a virtual compute resource.  required: false |

#### 6.2.56.4 Examples

See clause 6.8.3.8.

#### 6.2.56.5 Additional Requirements

None.

### 6.2.57 tosca.datatypes.nfv.KvpData

#### 6.2.57.1 Description

The KvpData data type describes a set of key-value pairs information used to customize a virtualised compute resource by using only key-value pairs data. Table 6.2.57.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.57.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | KvpData |
| **Type Qualified Name** | toscanfv:KvpData |
| **Type URI** | tosca.datatypes.nfv.KvpData |

#### 6.2.57.2 Properties

The properties of the MetaData data type shall comply with the provisions set out in table 6.2.57.2-1.

Table 6.2.57.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| data | no | map of string |  | A map of strings that contains a set of key-value pairs that describes the information for configuring the virtualised compute resource. |

#### 6.2.57.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.KvpData:  derived\_from: tosca.datatypes.Root  description: describes a set of key-value pairs information used to customize a virtualised compute resource at boot time by using only key-value pairs data.  properties:  data:  type: map  description: A map of strings that contains a set of key-value pairs that describes the information for configuring the virtualised compute resource.  required: false  entry\_schema:  type: string |

#### 6.2.57.4 Examples

See clause 6.8.3.8.

#### 6.2.57.5 Additional Requirements

None.

### 6.2.58 tosca.datatypes.nfv.ContentOrFileData

#### 6.2.58.1 Description

The ContentOrFileData data type describes a string content or a file information used to customize a virtualised compute resource by using string content or file. Table 6.2.58.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.58.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | ContentOrFileData |
| **Type Qualified Name** | toscanfv:ContentOrFileData |
| **Type URI** | tosca.datatypes.nfv.ContentOrFileData |

#### 6.2.58.2 Properties

The properties of the ContentOrFileData data type shall comply with the provisions set out in table 6.2.58.2-1.

Table 6.2.58.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| data | no | map of string |  | A map of strings that contains a set of key-value pairs that carries the dynamic deployment values which used to replace the corresponding variable parts in the file as identify by a URL as described in source\_path. Shall be present if "source\_path" is present and shall be absent otherwise. See note 1. |
| content | no | string |  | The string information used to customize a virtualised compute resource at boot time.  See note 2. |
| source\_path | no | string |  | The URL to a file contained in the VNF package used to customize a virtualised compute resource. The content shall comply with IETF RFC 3986 [8].  See note 2. |
| destination\_path | no | string |  | The URL where to inject a file indicated by the source\_path property into the virtualised compute resource. The content shall comply with IETF RFC 3986 [8].  See note 3. |
| NOTE 1: It is the file processor (e.g. in the VNFM) responsibility to replace the corresponding variable parts in the file with the value carried in the data property, the variable parts in the file are start with $ and end with $, its content is the same character with one of the keys in the data property, for example, if one of the keys in 'data' is "https\_proxy", somewhere in the file content there is $https\_proxy$.  NOTE 2: One and only one of the following properties shall be present: contents or source\_path.  NOTE 3: It is only present when a particular method is used for transferring boot information into a virtualised compute resource and source\_path is also present. For example, such method can be the personality method as described in [i.15], and it has been deprecated since Openstack 12.0.0 (Stein). | | | | |

#### 6.2.58.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.ContentOrFileData:  derived\_from: tosca.datatypes.Root  description: describes a string content or a file information used to customize a virtualised compute resource at boot time by using string content or file.  properties:  data:  type: map  description: A map of strings that contains a set of key-value pairs that carries the dynamic deployment values which used to replace the corresponding variable parts in the file as identify by a URL as described in source\_path. Shall be present if "source\_path" is present and shall be absent otherwise..  required: false  entry\_schema:  type: string  content:  type: string  description: The string information used to customize a virtualised compute resource at boot time.  required: false  source\_path:  type: string  description: The URL to a file contained in the VNF package used to customize a virtualised compute resource. The content shall comply with IETF RFC 3986 [8].  required: false  destination\_path:  type: string  description: The URL address when inject a file into the virtualised compute resource. The content shall comply with IETF RFC 3986 [8].  required: false |

#### 6.2.58.4 Examples

See clause 6.8.3.8.

#### 6.2.58.5 Additional Requirements

None.

### 6.2.59 tosca.datatypes.nfv.BootDataVimSpecificProperties

#### 6.2.59.1 Description

The BootDataVimSpecificProperties data type describes the VIM related information used for selecting VIM specific capabilities when setting the boot data when setting the boot data. Table 6.2.59.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.2.59.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | BootDataVimSpecificProperties |
| **Type Qualified Name** | toscanfv: BootDataVimSpecificProperties |
| **Type URI** | tosca.datatypes.nfv.BootDataVimSpecificProperties |

#### 6.2.59.2 Properties

The properties of the BootDataVimSpecificProperties data type shall comply with the provisions set out in table 6.2.59.2-1.

Table 6.2.59.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| vim\_type | no | string |  | Discriminator for the different types of the VIM information.  The set of permitted values is expected to change over time as new types or versions of VIMs become available.  The ETSI NFV registry of VIM-related information [i.16] provides access to information about various VIM types. |
| properties | yes | map of string |  | Properties used for selecting VIM specific capabilities when setting the boot data. For example, it can set whether config\_drive functionality is selected in case VIM support it. This property is reserved for future use in the present document. |

#### 6.2.59.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.BootDataVimSpecificProperties:  derived\_from: tosca.datatypes.Root  description: describes the VIM specific information used for selecting VIM specific capabilities when setting the boot data.  properties:  vim\_type:  type: string  description: Discriminator for the different types of the VIM information.  required: true  properties:  type: map  description: Properties used for selecting VIM specific capabilities when setting the boot data  entry\_schema:  type: string  required: true |

#### 6.2.59.4 Examples

None.

#### 6.2.59.5 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** | SwImage |
| **Type Qualified Name** | toscanfv:SwImage |
| **Type URI** | tosca.artifacts.nfv.SwImage |

#### 6.3.1.2 Description

|  |
| --- |
| tosca.artifacts.nfv.SwImage:  derived\_from: tosca.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** | Mistral |
| **Type Qualified Name** | toscanfv:Mistral |
| **Type URI** | tosca.artifacts.Implementation.nfv.Mistral |
| **derived\_from** | tosca.artifacts.Implementation |

#### 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: tosca.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** | VirtualBindable |
| **Type Qualified Name** | toscanfv:VirtualBindable |
| **Type URI** | tosca.capabilities.nfv.VirtualBindable |

#### 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: tosca.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

The VirtualLinkable capability type is defined in clause 9.4.1 of the present document.

### 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 virtualised compute. |
| virtual\_cpu | yes | tosca.datatypes.nfv.VirtualCpu |  | Describes virtual CPU(s) of the virtualised 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: tosca.datatypes.nfv.LogicalNodeData  requested\_additional\_capabilities:  type: map  description: Describes additional capability for a particular VDU  required: false  entry\_schema:  type: tosca.datatypes.nfv.RequestedAdditionalCapability  compute\_requirements:  type: map  required: false  entry\_schema:  type: string  virtual\_memory:  type: tosca.datatypes.nfv.VirtualMemory  description: Describes virtual memory of the virtualised compute  required: true  virtual\_cpu:  type: tosca.datatypes.nfv.VirtualCpu  description: Describes virtual CPU(s) of the virtualised 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: tosca.datatypes.nfv.VirtualBlockStorageData  description: virtual system disk definition |

### 6.4.4 tosca.capabilities.nfv.VirtualStorage

#### 6.4.4.1 Description

The VirtualStorage capability indicates that the node that includes it can be pointed by a tosca.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: tosca.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** | VirtualBindsTo |
| **Type Qualified Name** | toscanfv:VirtualBindsTo |
| **Type URI** | tosca.relationships.nfv.VirtualBindsTo |

#### 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: tosca.relationships.DependsOn  description: Represents an association relationship between Vdu.Compute and VduCp node types  valid\_target\_types: [ tosca.capabilities.nfv.VirtualBindable ] |

### 6.6.2 tosca.relationships.nfv.VirtualLinksTo

#### 6.6.2.1 Description

The VirtualLinksTo relationship type is defined in clause 9.6.1 of the present document representing an association relationship between a VduCp and a VnfVirtualLink node type or a VnfExtCp and a VnfVirtualLink node type.

### 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 Flavour
* Change 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: tosca.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: tosca.datatypes.nfv.VnfOperationAdditionalParameters  # required: false  # derived types are expected to introduce additional\_parameters with its  # type derived from tosca.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: tosca.datatypes.nfv.VnfOperationAdditionalParameters  # required: false  # derived types are expected to introduce additional\_parameters with its  # type derived from tosca.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: tosca.datatypes.nfv.VnfOperationAdditionalParameters  # required: false  # derived types are expected to introduce additional\_parameters with its  # type derived from tosca.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 tosca.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 6.2.43 (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 6.3.2 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, etc.) 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: tosca.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]).

### 6.7.2 tosca.interfaces.nfv.VnfIndicator

#### 6.7.2.1 Description

The tosca.interfaces.nfv.VnfIndictor is an empty base interface type for deriving VNF specific interface types that include VNF indicator specific notifications.

Table 6.7.2.1-1 specifies the declared names for this interface type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.3 [20].

Table 6.7.2.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VnfIndicator |
| **Type Qualified Name** | toscanfv:VnfIndicator |
| **Type URI** | tosca.interfaces.nfv.VnfIndicator |

#### 6.7.2.2 Definition

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

|  |
| --- |
| tosca.interfaces.nfv.VnfIndicator:  derived\_from: tosca.interfaces.Root  description: This interface is an empty base interface type for deriving VNF specific interface types that include VNF indicator specific notifications. |

#### 6.7.2.3 Examples

See clause 6.8.1.9.

## 6.8 Node Types

### 6.8.1 tosca.nodes.nfv.VNF

#### 6.8.1.1 Description

The VNF node type is the generic abstract type from which all VNF specific 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.  See note 3.  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. |
| 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.datatypes.nfv.VnfConfigurableProperties |  | Describes the configurable properties of the VNF (e.g. related to auto scaling and auto healing). |
| modifiable\_attributes | no | tosca.datatypes.nfv.VnfInfoModifiableAttributes |  | Describes the modifiable attributes of the VNF. |
| lcm\_operations\_configuration | no | tosca.datatypes.nfv.VnfLcmOperationsConfiguration |  | Describes the configuration parameters for the VNF LCM operations. |
| monitoring\_parameters | no | list of tosca.datatypes.nfv.VnfMonitoringParameter |  | Describes monitoring parameters applicable to the VNF.  See note 4. |
| 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.datatypes.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 an NSD service template when describing a VNF node template with the corresponding VnfProfile information.  NOTE 3: The value of the descriptor\_id string shall comply with an UUID format as specified in section 3 of IETF RFC 4122 [9].  NOTE 4: This property is only used in a VNFD service template when describing a VNF node template with the corresponding monitoring information. | | | | |

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 items 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 = "."

This implies that vnfm\_info string values shall also comply with the pattern defined by the following regular expression [15]: (^etsivnfm:v[0-9]?[0-9]\.[0-9]?[0-9]\.[0-9]?[0-9]$)|(^[0-9]+:[a-zA-Z0-9.-]+$)

#### 6.8.1.3 Attributes

If the VNF supports VNF indicators, the VNF node type definition shall include one TOSCA attribute of a primitive type for each supported VNF indicator. The attribute definition shall indicate all possible values as constraints.

NOTE: In this version of the present document, the type of VNF indicators is constrained to primitive types. This is due to the limitations in the TOSCA-Simple-Profile-YAML-v1.3 [20] to define conditions on attributes of complex types.

VNF indicators may be defined in the VNFD to allow for the asynchronous notification of VNF specific information to the VNFM.

An attribute defined in the VNF node type for a VNF indicator holds the value for that indicator during the lifecycle of the VNF. A notification defined in the derived interface for VNF indicators (see clause 6.7.2) produces an output value which is assigned to the attribute, as per TOSCA Simple Profile in YAML 1.3 syntax. Examples of such assignments are shown in clause 6.8.1.9. Thus, the value of the VNF indicator may change every time a notification is received.

#### 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 | Capability type | Constraints | Description |
| --- | --- | --- | --- | --- |
| virtual\_link | no | 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 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 # UUID  description: Identifier of this VNFD information element. This attribute shall be globally unique  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  constraints:  - pattern: (^etsivnfm:v[0-9]?[0-9]\.[0-9]?[0-9]\.[0-9]?[0-9]$)|(^[0-9]+:[a-zA-Z0-9.-]+$)  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: tosca.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 tosca.datatypes.nfv.VnfConfigurableProperties  #modifiable\_attributes:  #type: tosca.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  # tosca.datatypes.nfv.VnfInfoModifiableAttributes  lcm\_operations\_configuration:  type: tosca.datatypes.nfv.VnfLcmOperationsConfiguration  description: Describes the configuration parameters for the VNF LCM operations  required: false  monitoring\_parameters:  type: list  entry\_schema:  type: tosca.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: tosca.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: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [ 0, 1 ]  # Additional requirements shall be defined in the VNF specific node type (deriving from tosca.nodes.nfv.VNF) corresponding to NS virtual links that need to connect to VnfExtCps  interfaces:  Vnflcm:  type: tosca.interfaces.nfv.Vnflcm  # VnfIndicator:  # type: tosca.interfaces.nfv.VnfIndicator  # derived types are expected to introduce Vnf Indicator interfaces  # with their type derived from tosca.interfaces.nfv.VnfIndicator |

#### 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:

1. The node type shall be derived from: tosca.nodes.nfv.VNF.
2. The following properties listed in tosca.nodes.nfv.VNF where the "required:" field is set to "true" shall be included with their values indicated as constraints and as default values:
   1. descriptor\_id
   2. descriptor\_version
   3. provider
   4. product\_name
   5. software\_version
   6. vnfm\_info
   7. flavour\_id
   8. flavour\_description

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

1. An empty string shall be indicated as the default value of the flavour\_description property.
2. The capabilities, requirements, interfaces of tosca.nodes.nfv.VNF shall be preserved.
3. 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 with the occurrences set to [ 0, 1 ]. In this case, it is the VNFD author's choice to use the requirement for VirtualLinkable capability defined in the tosca.nodes.nfv.VNF node type or use only the additional requirements defined in the derived VNF specific node type. In the latter case, the virtual\_link requirement should be included in the node type definition with occurrences [ 0, 0 ].

If the external connection point exposes a VipCp, a new requirement for VirtualLinkableCapability using the VipVirtualLinksTo relationship shall be defined for this connection point.

1. 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. "\_").

1. If the VNF supports VNF indicators, the VNF node type definition shall include an interface definition of a VNF specific interface type indicating the mapping of notification outputs to the VNF node attributes. For each of the VNF indicators, the name of the notification output shall be the same as the name of the corresponding VNF attribute.

NOTE 3: The notifications keyname in TOSCA interface is defined in TOSCA-Simple-Profile-YAML-v1.3 [20].

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

* tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions.
* tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata.
* tosca.datatypes.nfv.VnfAdditionalConfigurableProperties.
* tosca.datatypes.nfv.VnfInfoModifiableAttributes.
* tosca.datatypes.nfv.VnfConfigurableProperties.

See illustrative examples in clauses 6.8.1.9, 6.2.35.4 and 6.2.31.4.

In the derived VNF node type, the modifiable\_attributes and configurable\_properties (VNF-specific extension of the tosca.datatypes.nfv.VnfInfoModifiableAttributes and tosca.datatypes.nfv.VnfConfigurableProperties, respectively, by extending the above listed types) describe the name and type of the modifiable attributes (extensions and metadata) and configurable properties (vnfConfigurableProperties).

The modifiable\_attributes and configurable\_properties information provided in the node type is sufficient for the client of the VNF LCM API for providing values to these properties. A value provided via the VNF LCM API to such a property overrides the value (if any) assigned in the node template or defined as default value in the node type definition.

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.

For a given NSD, when describing a referenced VNFD as a node templates, the monitoring\_parameters property shall not be included.

#### 6.8.1.9 Example

Example usage of modifiable\_attributes:

|  |
| --- |
| tosca\_definitions\_version: tosca\_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:  extensions:  type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions  mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions:  derived\_from: tosca.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: tosca\_simple\_yaml\_1\_2  node\_types:  MyCompany.SunshineDB.1\_0.1\_0:  derived\_from: tosca.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 an 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 |

Example usage of VNF indicators attributes in VNF node type definition and VNF indicator notifications in interface definition:

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3  interface\_types:  tosca.nfv.interfaces.MyCompanyVnfIndicator  derived\_from: tosca.nfv.interfaces.VnfIndicator  notifications:  health:  description: this notification is used to received asynchronous information of value change of the health\_vnf\_indicator  utilization:  description: this notification is used to received asynchronous information of value change of the utilization\_vnf\_indicator  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  attributes:  health\_vnf\_indicator:  type: string  constraints:  - valid\_values: [ green, red, yellow ]  utilization\_vnf\_indicator:  type: float  constraints:  - in\_range [ 0.0, 100.0 ]  interfaces:  Vnflcm:  type: tosca.interfaces.nfv.Vnflcm  operations:  # omitted for brevity  VnfIndicator:  type: tosca.interfaces.nfv.MyCompanyVnfIndicator  notifications:  health:  output:   health\_vnf\_indicator: [ SELF, health\_vnf\_indicator ]  utilization:  output:   utilization\_vnf\_indicator: [ SELF, utilization\_vnf\_indicator ] |

### 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** | VnfExtCp |
| **Type Qualified Name** | toscanfv:VnfExtCp |
| **Type URI** | tosca.nodes.nfv.VnfExtCp |

#### 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 tosca.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 | Capability 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 Capabilities

None.

#### 6.8.2.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.VnfExtCp:  derived\_from: tosca.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: tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements  requirements:  - external\_virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [0, 1]  - internal\_virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [1, 1] |

#### 6.8.2.7 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.

#### 6.8.2.8 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 a VNF specific node type. In this substitution, the virtual\_link requirement is mapped to the external\_virtual\_link requirement of the VnfExtCp node. This example is illustrated in clause A.3.3.

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 a VNF specific node type is substituted by a service template representing a certain VNF deployment flavour. In this substitution, the virtual\_link requirement is mapped to the virtual\_link requirement of the VduCp node. This example is illustrated in clause A.3.2.

### 6.8.3 tosca.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 | yes | boolean |  | It indicates whether the order of the virtual\_storage requirements is used as the boot index (the first requirement represents the lowest index and defines highest boot priority).  If no boot order is indicated or the value is false, the default boot order defined in the VIM or NFVI shall be used. |
| nfvi\_constraints | no | map 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.  This property is reserved for future use in the present document. |
| monitoring\_parameters | no | list of tosca.datatypes.nfv.VnfcMonitoringParameter |  | Describes monitoring parameters applicable to a VNFC based on this VDU. |
| configurable\_properties | no | map of tosca.datatypes.nfv.VnfcConfigurableProperties |  | Describes the configurable properties of all VNFC instances based on this VDU. |
| boot\_data | no | tosca.datatypes.nfv.BootData |  | Contains the information 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 | Capability type | Constraints | Description |
| --- | --- | --- | --- | --- |
| virtual\_storage | no | tosca.capabilities.nfv.VirtualStorage |  | Describes storage requirements for a virtual\_storage instance attached to the virtualisation 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: tosca.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: boolean  description: indicates whether the order of the virtual\_storage requirements is used as the boot index (the first requirement represents the lowest index and defines highest boot priority)  required: true  default: false  nfvi\_constraints:  type: map  description: Describes constraints on the NFVI for the VNFC instance(s) created from this VDU. This property is reserved for future use in the present document.  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: tosca.datatypes.nfv.VnfcMonitoringParameter  #configurable\_properties:  #type: tosca.datatypes.nfv.VnfcConfigurableProperties  #required: false  # derived types are expected to introduce  # configurable\_properties with its type derived from  # tosca.datatypes.nfv.VnfcConfigurableProperties  vdu\_profile:  type: tosca.datatypes.nfv.VduProfile  description: Defines additional instantiation data for the VDU.Compute node  required: true  sw\_image\_data:  type: tosca.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 tosca.artifacts.nfv.SwImage and not required otherwise  boot\_data:  type: tosca.datatypes.nfv.BootData  description: Contains the information 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. There shall be a maximum number of one such artifact definition in a tosca.nodes.nfv.Vdu.Compute node template. The node template name of type tosca.nodes.nfv.Vdu.Compute fulfills the purpose of the "SwImageDescId" attribute of the SwImageDesc information element in ETSI GS NFV-IFA 011 [1] and hence it will be used in APIs to identify the software image id from the VNFD perspective. See example in clause 6.8.3.8.

When VNF-specific configurable properties are defined at the VDU-level, VNF providers shall define a VNF/VDU specific Vdu.Compute node type, where the configurable\_properties property has a datatype derived from tosca.datatypes.nfv.VnfcConfigurableProperties. See example in clause 6.2.10.4.

The VNF/VDU specific Vdu.Compute node type shall be defined as follows:

* All properties listed in tosca.nodes.nfv.Vdu.Compute where the "required:" field is set to "true" shall be included.
* The capabilities and requirements of tosca.nodes.nfv.Vdu.Compute shall be preserved.
* The configurable\_properties property shall have a datatype derived from tosca.datatypes.nfv.VnfcConfigurableProperties, according to the rules defined in clause 5.7.2 of the present document.

The definition of a VNF/VDU specific node type shall be included in one of the following yaml files:

1. In the yaml file which contains the corresponding VNF specific node type definition.
2. In low-level service templates or in the single TOSCA service template representing the VNFD in case of a single deployment flavour design with a single TOSCA service template.
3. In a standalone yaml file, to be imported from the low-level TOSCA service templates or from the single TOSCA service template representing the VNFD in case of a single deployment flavour design with a single TOSCA service template.
4. In any other VNF-specific files containing type definitions used by the VNFD TOSCA service template.

In the derived Vdu.Compute node type, the additional\_vnfc\_configurable\_properties (VNF/VDU-specific extension of the tosca.datatypes.nfv.VnfAdditionalVnfcConfigurableProperties data type and the containing tosca.datatypes.nfv.VnfcConfigurableProperties data type) describe the name and type of the VNFC configurable properties.

The additional\_vnfc\_configurable\_properties information provided in the node type is sufficient for the client of the VNF LCM API for providing values to these properties. A value provided via the VNF LCM API to such a property overrides the value (if any) assigned in the node template or defined as default value in the node type definition.

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] modelling 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 containing kvp\_data.

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

ip\_address\_1:

type: string

required: false

vm\_Nname:

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: tosca.nodes.nfv.Vdu.Compute

properties:

..

boot\_data:

kvp\_data:

data:

ip\_address\_1: { get\_property: [vnf, modifiable\_attributes, extensions, ip\_address\_1 ] }

..

This example illustrates fetching the boot data value by using content\_or\_file\_data.

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

ip\_address\_1:

type: string

required: false

vm\_name:

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: tosca.nodes.nfv.Vdu.Compute

properties:

..

boot\_data:

content\_or\_file\_data

contents: { 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 by using content\_or\_file\_data.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

..

topology\_template:

..

node\_templates:

dbBackend:

type: tosca.nodes.nfv.Vdu.Compute

properties:

..

boot\_data:

content\_or\_file\_data

data:

http\_proxy: { get\_property: [vnf, modifiable\_attributes, extensions, http\_proxy ] }

source\_path: { get\_artifact : [ SELF, boot\_data ] }

artifacts:

sw\_image:

type: tosca.artifacts.nfv.SwImage

file: images/dbBackend.v1.0.1.qcow2

boot\_data:

type: tosca.artifacts.example

file: implementation/templates/boot\_data.file

..

This example illustrates fetching the boot data value from a file by using content\_or\_file\_data with destination\_path.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

..

topology\_template:

..

node\_templates:

dbBackend:

type: tosca.nodes.nfv.Vdu.Compute

properties:

..

boot\_data:

content\_or\_file\_data:

data:

vm\_name: get\_property: [vnf, modifiable\_attributes, extensions, vm\_name ]

source\_path: { get\_artifact : [ SELF, boot\_data ] }

destination\_path: /etc/

artifacts:

sw\_image:

type: tosca.artifacts.nfv.SwImage

file: images/dbBackend.v1.0.1.qcow2

boot\_data:

type: tosca.artifacts.example

file: implementation/templates/boot\_data.file

..

This example illustrates the association of a software image artifact to a Vdu.Compute node. The name of the artifact "dbBackend" will be used in external APIs to identify the image.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

..

topology\_template:

..

node\_templates:

dbBackend:

type: tosca.nodes.nfv.Vdu.Compute

properties:

..

sw\_image\_data:

..

artifacts:

sw\_image:

type: tosca.artifacts.nfv.SwImage

file: images/dbBackend.v1.0.1.qcow2

..

This example illustrates the association of a software image artifact to more than Vdu.Compute nodes. The name of the artifact "dbBackend and oamService" will be used in external APIs to identify the image of each Vdu.Compute node.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

..

topology\_template:

..

node\_templates:

dbBackend:

type: tosca.nodes.nfv.Vdu.Compute

properties:

..

sw\_image\_data:

..

artifacts:

sw\_image:

type: tosca.artifacts.nfv.SwImage

file: images/dbBackend.v1.0.1.qcow2

oamService:

type: tosca.nodes.nfv.Vdu.Compute

properties:

..

sw\_image\_data:

..

artifacts:

sw\_image:

type: tosca.artifacts.nfv.SwImage

file: images/dbBackend.v1.0.1.qcow2

..

### 6.8.4 tosca.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: tosca.nodes.Root  description: This node type describes the specifications of requirements related to virtual block storage resources  properties:  virtual\_block\_storage\_data:  type: tosca.datatypes.nfv.VirtualBlockStorageData  description: Describes the block storage characteristics.  required: true  sw\_image\_data:  type: tosca.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 tosca.artifacts.nfv.SwImage and not required otherwise  capabilities:  virtual\_storage:  type: tosca.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. There shall be a maximum number of one such artifact definition in a tosca.nodes.nfv.Vdu.VirtualBlockStorage node template. The node template name of type tosca.nodes.nfv.Vdu.VirtualBlockStorage fulfills the purpose of the "SwImageDescId" attribute of the SwImageDesc information element in ETSI GS NFV-IFA 011 [1] and hence it will be used in APIs to identify the software image id from the VNFD descriptor. See example in clause 6.8.3.8.

### 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** | VirtualObjectStorage |
| **Type Qualified Name** | toscanfv:VirtualObjectStorage |
| **Type URI** | tosca.nodes.nfv.Vdu.VirtualObjectStorage |

#### 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** |
| --- | --- | --- | --- | --- |
| virtual\_object\_storage\_data | yes | tosca.datatypes.nfv.VirtualObjectStorageData |  | 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: tosca.nodes.Root  description: This node type describes the specifications of requirements related to virtual object storage resources  properties:  virtual\_object\_storage\_data:  type: tosca.datatypes.nfv.VirtualObjectStorageData  description: Describes the object storage characteristics.  required: true  capabilities:  virtual\_storage:  type: tosca.capabilities.nfv.VirtualStorage  description: Defines the capabilities of virtual\_storage. |

#### 6.8.5.7 Additional requirements

None.

### 6.8.6 tosca.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 | Capability 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  occurrences: [1, 1]  # 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

The Cp node type is defined in clause 9.8.1 of the present document.

### 6.8.8 tosca.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.datatypes.nfv.VirtualNetworkInterfaceRequirements |  | Specifies requirements on a virtual network interface realizing 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 \*mandatory\* 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, macvtap, direct, baremetal,virtio-forwarder, direct-physical, smart-nic | 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 | Capability 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: tosca.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 realizing the CPs instantiated from this CPD  required: false  entry\_schema:  type: tosca.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, macvtap, direct, baremetal, virtio-forwarder, direct-physical, smart-nic ]  requirements:  - virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [0, 1]  - virtual\_binding:  capability: tosca.capabilities.nfv.VirtualBindable  relationship: tosca.relationships.nfv.VirtualBindsTo  node: tosca.nodes.nfv.Vdu.Compute  occurrences: [1, 1] |

### 6.8.9 tosca.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.VlProfile |  | 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: tosca.nodes.Root  description: Describes the information about an internal VNF VL  properties:  connectivity\_type:  type: tosca.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: tosca.datatypes.nfv.VlProfile  description: Defines additional data for the VL  required: true  monitoring\_parameters:  type: list  entry\_schema:  type: tosca.datatypes.nfv.VirtualLinkMonitoringParameter  description: Describes monitoring parameters applicable to the VL  required: false  capabilities:  virtual\_linkable:  type: tosca.capabilities.nfv.VirtualLinkable |

### 6.8.10 tosca.nodes.nfv.VipCp

#### 6.8.10.1 Description

A VipCp node type represents the VipCpd information element as defined in ETSI GS NFV-IFA 011 [1], which describes a connection point to allocate one or a set of virtual IP addresses. Table 6.8.10.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.8.10.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VipCp |
| **Type Qualified Name** | toscanfv:VipCp |
| **Type URI** | tosca.nodes.nfv.VipCp |

#### 6.8.10.2 Properties

The properties of the VipCp node type shall comply with the provisions set out in table 6.8.10.2-1.

Table 6.8.10.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| vip\_function | yes | string | valid values:  high\_availability,  load\_balance | Indicates the function the virtual IP address is used for: high availability or load balancing. See note. |
| NOTE: When used for high availability, only one of the internal VDU CP instances or VNF external CP instances that share the virtual IP is bound to the VIP address at a time, i.e. only one is configured in the external (to the VNF) router to receive the packets e.g. as a result of a G-ARP message previously sent by this instance. When used for load balancing purposes all CP instances that share the virtual IP are bound to it. A load balancing function sends the packet to one or the other, but not to both. | | | | |

#### 6.8.10.3 Attributes

None.

#### 6.8.10.4 Requirements

The requirements of the VipCp node type shall comply with the provisions set out in table 6.8.10.4-1.

Table 6.8.10.4-1: Requirements

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| target | yes | tosca.capabilities.Node |  | Describes the requirement for connecting to CP instances that share the virtual IP address. |
| virtual\_link | yes | tosca.capabilities.nfv.VirtualLinkable |  | Describes the requirements for linking to virtual link. |

#### 6.8.10.5 Definition

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

|  |
| --- |
| tosca.nodes.nfv.VipCp:  derived\_from: tosca.nodes.nfv.Cp  description: Describes a connection point to allocate one or a set of virtual IP addresses  properties:  vip\_function:  type: string  description: "Indicates the function the virtual IP address is used for: high availability or load balancing. When used for high availability, only one of the internal VDU CP instances or VNF external CP instances that share the virtual IP is bound to the VIP address at a time. When used for load balancing purposes all CP instances that share the virtual IP are bound to it."  required: true  constraints:  - valid\_values: [ high\_availability, load\_balance ]  requirements:  - target:  capability: tosca.capabilities.Node  relationship: tosca.relationships.DependsOn  occurrences: [ 1, UNBOUNDED ]  - virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VipVirtualLinksTo  occurrences: [1, 1] |

#### 6.8.10.6 Example

See clause A.13.

## 6.9 Group Types

### 6.9.1 tosca.groups.nfv.PlacementGroup

#### 6.9.1.1 Description

PlacementGroup is used for describing the affinity or anti-affinity relationship applicable between the virtualisation 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: tosca.groups.Root  description: PlacementGroup is used for describing the affinity or anti-affinity relationship applicable between the virtualisation 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** | InstantiationLevels |
| **Type Qualified Name** | toscanfv:InstantiationLevels |
| **Type URI** | tosca.policies.nfv.InstantiationLevels |

#### 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 |
| --- | --- | --- | --- | --- |
| levels | yes | map of tosca.datatypes.nfv.InstantiationLevel |  | Describes the various levels of resources that can be used to instantiate the VNF using this flavour. |
| default\_level | 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  constraints:  - min\_length: 1  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.datatypes.nfv.VduLevel |  | 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  constraints:  - min\_length: 1  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  constraints:  - min\_length: 1  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  constraints:  - min\_length: 1 |

#### 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** | VduScalingAspectDeltas |
| **Type Qualified Name** | toscanfv:VduScalingAspectDeltas |
| **Type URI** | tosca.policies.nfv.VduScalingAspectDeltas |

#### 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** |
| --- | --- | --- | --- | --- |
| aspect: | yes | string |  | Represents the scaling aspect to which this policy applies. |
| deltas | yes | map of tosca.datatypes.nfv.VduLevel |  | Describes the Vdu.Compute 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: tosca.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: tosca.datatypes.nfv.VduLevel  constraints:  - min\_length: 1  targets: [ tosca.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  constraints:  - min\_length: 1  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: tosca.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: tosca.datatypes.nfv.VirtualLinkBitrateLevel  description: Represents the initial minimum size of the VNF.  required: true  targets: [ tosca.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 tosca.nodes.nfv.Vdu.Compute or tosca.nodes.nfv.VnfVirtualLink set as the targets, the AffinityRule or AntiAffinityRule applies between the virtualisation 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 tosca.nodes.nfv.Vdu.Compute or tosca.nodes.nfv.VnfVirtualLink 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 the virtualisation 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].

Tables 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 tosca.policies.nfv.Abstract.SecurityGroupRule

#### 6.10.11.1 Description

The Abstract.SecurityGroupRule policy type is defined in clause 9.10.1 of the present document.

### 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: tosca.datatypes.nfv.InterfaceDetails  description: Provide additional data to access the interface endpoint  required: false  targets: [ tosca.nodes.nfv.VnfExtCp, tosca.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: tosca.policies.nfv.SupportedVnfInterface  targets: [ MyVnfmFacingExtCp ]  properties:  interface\_name: vnf\_indicator  details:  uri\_components:  scheme: https  authority:  host: myvnf.example.com |

### 6.10.13 tosca.policies.nfv.SecurityGroupRule

#### 6.10.13.1 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 6.10.13.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 6.10.13.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | SecurityGroupRule |
| **Type Qualified Name** | toscanfv:SecurityGroupRule |
| **Type URI** | tosca.policies.nfv.SecurityGroupRule |

#### 6.10.13.2 Properties

None.

#### 6.10.13.3 targets

The targets of the SecurityGroupRule policy types shall comply with the provisions set out in table 6.10.13.3-1.

Table 6.10.13.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string[] | Valid types:  tosca.nodes.nfv.VduCp, tosca.nodes.nfv.VnfExtCp | Target connection points of VduCp and/or VnfExtCp |

#### 6.10.13.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.SecurityGroupRule:  derived\_from: tosca.policies.nfv.Abstract.SecurityGroupRule  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].  targets: [ tosca.nodes.nfv.VduCp, tosca.nodes.nfv.VnfExtCp ] |

#### 6.10.13.5 Additional Requirements

None.

## 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 VNF 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. A separate YAML file with a VNF specific node type definition which shall be derived from tosca.nodes.nfv.VNF node type as defined in clause 6.8.1 shall be provided and is also considered as a part of a VNFD. The top level service template shall be the main entry point of the VNF package as specified in ETSI GS NFV-SOL 004 [i.6], i.e. the Entry-definitions file.

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:

1. an import statement referencing the TOSCA types definition file as defined in clause B.2;
2. an import statement referencing the definitions of all low level service template(s);
3. an import statement referencing a yaml file which contains a VNF specific node type definition;
4. Optionally, import statements referencing additional VNF-specific files containing only type definitions used by this TOSCA service template; and
5. 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:

1. an import statement referencing the TOSCA types definition file as defined in clause B.2;
2. 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;
3. Optionally, import statements referencing additional VNF-specific files containing type definitions used by this TOSCA service template; and
4. a topology template describing the internal topology of the VNF with:

* substitution\_mappings indicating:
* the same node type as defined in the VNF specific node type definition 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 (or a derived node type), Vdu.VirtualBlockStorage, Vdu.VirtualObjectStorage, Vdu.VirtualFileStorage VduCp, etc. that define the topology and composition of the VNF flavour.

NOTE 1: The format and structure of a VNF package is defined in ETSI GS NFV-SOL 004 [i.6].

NOTE 2: All the imported type definition files as indicated either in the top level service template or in any of the lower level service template are considered as parts of a VNFD.

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 VNF package 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 VNFD with a single TOSCA service template.

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:

1. an import statement referencing the TOSCA types definition file as defined in clause B.2;
2. either a VNF specific node type definition derived from the tosca.nodes.nfv.VNF node type, as defined in clause 6.8.1 or an import statement referencing a file that contains such definition;
3. Optionally, import statements referencing additional VNF-specific files containing type definitions used by this TOSCA service template; and

d) a topology template describing the internal topology of the VNF with:

- substitution\_mappings indicating the same VNF specific node type and the mapping of the virtual\_link requirements on external connection points;

- a node template of this VNF specific 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 (or a derived node type), 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 VNF specific node type, which may appear in an NSD.

NOTE 3: All the imported type definition files as indicated in the service template are considered as parts of a VNFD.

# 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:

* NS scaling Aspect
* Multiple Ns instantiation levels
* VnfIndicators

Table 7.1-1: 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 |
| 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 |
| Vnffgd | tosca.groups.nfv.VNFFG | tosca.groups.Root |

## 7.2 Data Types

### 7.2.1 tosca.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 ETSI GS NFV-REL 001 [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: tosca.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 tosca.datatypes.nfv.VnfProfile

#### 7.2.2.1 Description

The VnfProfile data type is defined in clause 9.2.8 of the present document.

### 7.2.3 tosca.datatype.nfv.NsVlProfile

#### 7.2.3.1 Description

The NsVlProfile 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** | NsVlProfile |
| **Type Qualified Name** | toscanfv:NsVlProfile |
| **Type URI** | tosca.datatypes.nfv.NsVlProfile |

#### 7.2.3.2 Properties

The properties of the NsVlProfile 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 NsVlProfile is contained in the NsVirtualLink node. | | | | |

#### 7.2.3.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.NsVlProfile:  derived\_from: tosca.datatypes.Root  description: Describes additional instantiation data for a given NsVirtualLink used in a specific NS 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.NsVirtualLinkQos  description: Specifies the QoS requirements of a VL instantiated according to this profile.  required: false  service\_availability:  type: tosca.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 tosca.datatypes.nfv.ConnectivityType

#### 7.2.4.1 Description

The ConnectivityType data type is defined in clause 9.2.4 of the present document.

### 7.2.5 tosca.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 | greater\_or\_equal: 0 | 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: tosca.datatypes.nfv.Qos  description: describes QoS data for a given VL used in a VNF deployment flavour  properties:  priority:  type: integer  constraints:  - greater\_or\_equal: 0  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 9.2.5 of the present document.

### 7.2.7 Void

### 7.2.8 Void

### 7.2.9 Void

### 7.2.10 Void

### 7.2.11 tosca.datatypes.nfv.CpProtocolData

#### 7.2.11.1 Description

The CpProtocolData data type is defined in clause 9.2.6 of the present document.

### 7.2.12 tosca.datatypes.nfv.AddressData

#### 7.2.12.1 Description

The AddressData data type is defined in clause 9.2.3 of the present document.

### 7.2.13 tosca.datatypes.nfv.L2AddressData

#### 7.2.13.1 Description

The L2AddressData data type is defined in clause 9.2.1 of the present document.

### 7.2.14 tosca.datatypes.nfv.L3AddressData

#### 7.2.14.1 Description

The L3AddressData data type is defined in clause 9.2.2 of the present document.

### 7.2.15 tosca.datatypes.nfv.Qos

#### 7.2.15.1 Description

The Qos data type is defined in clause 9.2.7 of the present document.

### 7.2.16 tosca.datatypes.nfv.NsProfile

#### 7.2.16.1 Description

The NsProfile data type describes a profile for instantiating nested NSs which are constituents of an NS with a particular NS DF as defined in ETSI GS NFV-IFA 014 [2]. Table 7.2.16.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.16.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsProfile |
| **Type Qualified Name** | toscanfv:NsProfile |
| **Type URI** | tosca.datatypes.nfv.NsProfile |

#### 7.2.16.2 Properties

The properties of the NsProfile data type shall comply with the provisions set out in table 7.2.16.2-1.

Table 7.2.16.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| ns\_instantiation\_level | no | string |  | Identifier of the instantiation level of the NS DF to be used for instantiation. If not present, the default instantiation level as declared in the NSD shall be used. |
| min\_number\_of\_instances | yes | integer | greater\_or\_equal  : 0 | Minimum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile. |
| max\_number\_of\_instances | yes | integer | greater\_or\_equal  : 0 | Maximum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile. |
| flavour\_id | yes | string |  | Identifies the applicable network service DF within the scope of the NSD. |

#### 7.2.16.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.NsProfile:  derived\_from: tosca.datatypes.Root  description: describes a profile for instantiating NSs of a particular NS DF according to a specific NSD and NS DF.  properties:  ns\_instantiation\_level:  type: string  description: Identifier of the instantiation level of the NS DF to be used for instantiation. If not present, the default instantiation level as declared in the NSD shall be used.  required: false  min\_number\_of\_instances:  type: integer  description: Minimum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile.  required: true  constraints:  - greater\_or\_equal: 0  max\_number\_of\_instances:  type: integer  description: Maximum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile.  required: true  constraints:  - greater\_or\_equal: 0  flavour\_id:  type: string  description: Identifies the applicable network service DF within the scope of the NSD.  required: true |

#### 7.2.16.4 Example

None.

#### 7.2.16.5 Additional Requirements

None.

### 7.2.17 tosca.datatypes.nfv.Mask

#### 7.2.17.1 Description

The Mask data type describes the value to be matched for a sequence of bits at a particular location in a frame. Table 7.2.17.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.17.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | Mask |
| **Type Qualified Name** | toscanfv:Mask |
| **Type URI** | tosca.datatypes.nfv.Mask |

#### 7.2.17.2 Properties

The properties of the Mask data type shall comply with the provisions set out in table 7.2.17.2-1.

Table 7.2.17.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| starting\_point | yes | integer |  | Indicates the offset between the last bit of the source mac address and the first bit of the sequence of bits to be matched. |
| length | yes | integer |  | Indicates the number of bits to be matched. |
| value | yes | string |  | Provides the sequence of bit values to be matched. |

#### 7.2.17.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.Mask:  derived\_from: tosca.datatypes.Root  properties:  starting\_point:  description: Indicates the offset between the last bit of the source mac address and the first bit of the sequence of bits to be matched.  type: integer  required: true  length:  description: Indicates the number of bits to be matched.  type: integer  required: true  value:  description: Provide the sequence of bit values to be matched.  type: string  required: true |

#### 7.2.17.4 Examples

None.

### 7.2.18 tosca.datatypes.nfv.NsOperationAdditionalParameters

#### 7.2.18.1 Description

The NsOperationAdditionalParameters data type is an empty base type for deriving data types for describing NS specific additional parameters that affect the invocation of NS Lifecycle Management operations, as defined in ETSI GS NFV-IFA 014 [2]. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3]. Table 7.2.18.1-1 specifies the declared names for this data type.

Table 7.2.18.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsOperationAdditionalParameters |
| **Type Qualified Name** | toscanfv:NsOperationAdditionalParameters |
| **Type URI** | tosca.datatypes.nfv.NsOperationAdditionalParameters |

#### 7.2.18.2 Properties

None.

#### 7.2.18.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.NsOperationAdditionalParameters:  derived\_from: tosca.datatypes.Root  description: Is an empty base type for deriving data types for describing NS-specific additional parameters to be passed when invoking NS lifecycle management operations  #properties: |

#### 7.2.18.4 Examples

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2  node\_types:  tosca.example\_NS:  derived\_from: tosca.nodes.nfv.NS  properties:  ..  interfaces:  Nslcm:  instantiate:  inputs:  additional\_parameters:  type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters  scale:  inputs:  additional\_parameters:  type: MyCompany.datatypes.nfv.NsScaleAdditionalParameters  heal:  inputs:  additional\_parameters:  type: MyCompany.datatypes.nfv.NsHealAdditionalParameters  data\_types:  MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters:  derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters  properties:  parameter\_1:  type: string  required: true  parameter\_2:  type: string  required: true  default: value\_2  MyCompany.datatypes.nfv.NsScaleAdditionalParameters:  derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters  properties:  parameter\_1:  type: string  required: true  parameter\_2:  type: string  required: true  MyCompany.datatypes.nfv.NsHealAdditionalParameters:  derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters  properties:  parameter\_1:  type: string  required: true  parameter\_2:  type: string  required: true  default: value\_2 |

### 7.2.19 tosca.datatypes.nfv.NsMonitoringParameter

#### 7.2.19.1 Description

This data type is used to specify information on virtualised resource related performance metrics to be monitored at the NS level. Table 7.2.19.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.2.19.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| Shorthand Name | NsMonitoringParameter |
| Type Qualified Name | toscanfv:NsMonitoringParameter |
| Type URI | tosca.datatypes.nfv.NsMonitoringParameter |

#### 7.2.19.2 Properties

The properties of the NsMonitoringParameter data type shall comply with the provisions set out in table 7.2.19.2-1.

Table 7.2.19.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| name | no | string |  | Human readable name of the monitoring parameter. |
| performance\_metric | yes | string | valid values:  byte\_incoming\_sap  byte\_outgoing\_sap  packet\_incoming\_sap  packet\_outgoing\_sap  byte\_incoming  byte\_outgoing  packet\_incoming  packet\_outgoing | Identifies a performance metric to be monitored.  Performance metric values shall be either set to:   * A measurement name defined in clause 7.3 of ETSI GS NFV-IFA 027 [7]. In this case the NFVO computes these measurements from lower-level metrics collected from the VIM. * A measurement name defined in clause 7.1 of ETSI GS NFV-IFA 027 [7], without appending a sub-counter. In this case the NFVO collects these metrics from the VIM for all network resources allocated to all NS virtual links. |
| collection\_period | no | scalar-unit.time |  | Describes the periodicity at which to collect the performance information. |

#### 7.2.19.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.NsMonitoringParameter:  derived\_from: tosca.datatypes.Root  description: Represents information on virtualised resource related performance metrics applicable to the NS.  properties:  name:  type: string  description: Human readable name of the monitoring parameter  required: true  performance\_metric:  type: string  description: Identifies a performance metric to be monitored, according to ETSI GS NFV-IFA 027.  required: true  constraints:  - valid\_values: [byte\_incoming\_sap, byte\_outgoing\_sap, packet\_incoming\_sap, packet\_outgoing\_sap, byte\_incoming, byte\_outgoing, packet\_incoming, packet\_outgoing ]  collection\_period:  type: scalar-unit.time  description: Describes the periodicity at which to collect the performance information.  required: false |

#### 7.2.19.4 Examples

None.

#### 7.2.19.5 Additional Requirements

None.

### 7.2.20 tosca.datatypes.nfv.VnfMonitoringParameter

The VnfMonitoringParameter data type is defined in clause 9.2.9 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 9.4.1 of the present document.

### 7.4.2 tosca.capabilities.nfv.Forwarding

#### 7.4.2.1 Description

The Forwarding capability type describes the capabilities related to nodes which can be pointed by tosca.relationships.nfv.ForwardTo relationship type. Table 7.4.2.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

NOTE: The forwarding capability represents the ability of a CP or SAP to receive and forward traffic flows. Traffic flows can be received by a CP from an NS Virtual Link and forwarded to the VNF or PNF to which the CP is attached. Symmetrically, traffic flows can be received by an external CP of the VNF or PNF and forwarded to an NS Virtual Link. Traffic flows can be received by a SAP from an external link and forwarded to the NS to which the SAP is attached. Symmetrically, traffic flows can be received from the NS to which the SAP is attached and forwarded to an external link. An ingress CP is an external CP that forwards traffic to a VNF, PNF or NS while and egress CP is an external CP that forwards traffic outside a VNF, PNF or NS. The same CP may but need not play both roles.

Table 7.4.2.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | Forwarding |
| **Type Qualified Name** | toscanfv:Forwarding |
| **Type URI** | tosca.capabilities.nfv.Forwarding |

#### 7.4.2.2 Properties

None.

#### 7.4.2.3 Definition

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

|  |
| --- |
| tosca.capabilities.nfv.Forwarding:  derived\_from: tosca.capabilities.Root |

## 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 9.6.1 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 an NSD.

### 7.6.2 tosca.relationships.nfv.ForwardTo

#### 7.6.2.1 Description

The ForwardTo relationship type represents an association between two node types which are a part of NFP. Table 7.6.2.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.6.2.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | ForwardTo |
| **Type Qualified Name** | toscanfv:ForwardTo |
| **Type URI** | tosca.nodes.relationships.ForwardTo |

#### 7.6.2.2 Properties

None.

#### 7.6.2.3 Definition

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

|  |
| --- |
| tosca.relationships.nfv.ForwardTo:  derived\_from: tosca.relationships.Root  valid\_target\_types: [ tosca.capabilities.nfv.Forwarding ] |

## 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** | Nslcm |
| **Type Qualified Name** | toscanfv:Nslcm |
| **Type URI** | tosca.interfaces.nfv.Nslcm |

#### 7.7.1.2 Definition

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

|  |
| --- |
| tosca.interfaces.nfv.Nslcm:  derived\_from: tosca.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.  # inputs:  # additional\_parameters:  # type: tosca.datatypes.nfv.NsOperationAdditionalParameters  # required: false  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.  # inputs:  # additional\_parameters:  # type: tosca.datatypes.nfv.NsOperationAdditionalParameters  # required: false  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.  # inputs:  # additional\_parameters:  # type: tosca.datatypes.nfv.NsOperationAdditionalParameters  # required: false  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.

In the operation definitions on the Nslcm interface, the additional\_parameters (NS-specific extension of the tosca.datatypes.nfv.NsOperationAdditionalParameters) of inputs section describes the name and type of the additional parameters that can be submitted in the NS LCM operation request. Refer example in clause 7.2.17.

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 an NS LCM operation, the implementation logic is executed before or after the execution of the NS 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 tosca.nodes.nfv.NS

#### 7.8.1.1 Description

The NFV Network Service (NS) node type describes an 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.  See note 2. |
| 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. See note 2. |
| flavour\_id | yes | string |  | Identifier of this NS DF within the NSD. |
| ns\_profile | no | tosca.datatypes.nfv.NsProfile |  | Specifies a profile of an NS, when this NS is used as nested NS within another NS. See note 1. |
| NOTE 1: This property is only used in an NS node template, when it is representing a nested NS within another NS.  NOTE 2: The value of the descriptor\_id string shall comply with an UUID format as specified in section 3 of IETF RFC 4122 [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 | Capability type | Constraints | Description |
| --- | --- | --- | --- | --- |
| virtual\_link | no | tosca.capabilities.nfv.VirtualLinkable |  | 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: tosca.nodes.Root  properties:  descriptor\_id:  type: string # UUID  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: # UUID  type: string  description: Identifies an NSD in a version independent manner. This attribute is invariant across versions of NSD  required: true  flavour\_id:  type: string  description: Identifier of the NS Deployment Flavour within the NSD  required: true  ns\_profile:  type: tosca.datatypes.nfv.NsProfile  description: Specifies a profile of a NS, when this NS is used as nested NS within another NS.  required: false  requirements:  - virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  node: tosca.nodes.nfv.NsVirtualLink  occurrences: [ 0, 1 ]  interfaces:  Nslcm:  type: tosca.interfaces.nfv.Nslcm |

#### 7.8.1.7 Artifact

None.

#### 7.8.1.8 Additional requirements

For a given NSD, a new NS node type shall be defined following the below requirements:

1. The node type shall be derived from: tosca.nodes.nfv.NS.
2. All properties listed in tosca.nodes.nfv.NS where the "required:" field is set to "true" shall be included with their values indicated as constraints.
3. Properties listed in in tosca.nodes.nfv.NS where the "required:" field is set to "false" may be included.
4. The capabilities, requirements, interfaces of tosca.nodes.nfv.NS shall be preserved.
5. Depending on the number of SAPs of the NS, additional requirements for VirtualLinkable capability shall be defined with the occurrences set to [ 0, 1 ]. In this case, it is the NSD author's choice to use the requirement for VirtualLinkable capability defined in the tosca.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 tosca.nodes.nfv.Sap

#### 7.8.2.1 Description

The Service Access Point (SAP) node type describes a connection point where an 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 9.8.1 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 | Capability 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: tosca.nodes.nfv.Cp  description: node definition of SAP.  requirements:  - external\_virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [0, 1]  - internal\_virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [1, 1] |

#### 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 an NsVirtualLink inside an NSD. The node template has the following requirements:

* internal\_virtual\_link requirement to allow to connect it to an NsVirtualLink inside an NSD;
* external\_virtual\_link requirement to allow to connect it to an NsVirtualLink outside an 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 a NS specific node type. In this substitution, the virtual\_link requirement 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 a NS specific node type is substituted by a service template representing a certain NS deployment flavour. In this substitution, the virtual\_link requirement 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 |
| --- | --- | --- | --- | --- |
| vl\_profile | yes | tosca.datatype.nfv.NsVlProfile |  | Specifies instantiation parameters for a virtual link of a particular NS deployment flavour. |
| connectivity\_type | yes | tosca.datatypes.nfv.ConnectivityType |  | Network service virtual link connectivity type. |
| test\_access | no | list of string | Valid values: passive\_monitoring, active\_loopback | Test access facilities available on the VL. |
| description | 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: tosca.nodes.Root  description: node definition of Virtual Links  properties:  vl\_profile:  type: tosca.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: tosca.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: tosca.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 9.8.1 of the present document.

### 7.8.5 tosca.nodes.nfv.NfpPositionElement

#### 7.8.5.1 Description

The NfpPositionElement node type represents the NfpPositionElement information element as defined in ETSI GS NFV‑IFA 014 [2], which describes one or two CPD(s) or SAPD(s) for a given Vnf, Pnf or Ns. Table 7.8.5.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

A NfpPositionElement node type has a requirement for a forwarding capability to be exposed by the VNFs, PNFs or NSs, in order to re-expose this capability to an NfpPosition node type.

NOTE: The NfpPosition and NfpPositionElement node types of the VNFFG model describe the entities in VIM for enabling packets/frames to traverse the constituent VNFs, PNFs or Nested NSs of the Network Forwarding Path.

Table 7.8.5.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NfpPositionElement |
| **Type Qualified Name** | toscanfv:NfpPositionElement |
| **Type URI** | tosca.nodes.nfv.NfpPositionElement |

#### 7.8.5.2 Properties

None.

#### 7.8.5.3 Attributes

None.

#### 7.8.5.4 Requirements

The requirements of the NfpPositionElement node type shall comply with the provisions set out in table 7.8.5.4-1.

Table 7.8.5.4-1: Requirements

| **Name** | **Required** | **Capability Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| profile\_element | yes | tosca.capabilities.nfv.Forwarding |  | Describes the requirement for the constituent of the NfpPositionElement. |

#### 7.8.5.5 Capabilities

The capabilities of the NfpPositionElement node type shall comply with the provisions set out in table 7.8.5.5-1.

Table 7.8.5.5-1: Capabilities

| **Name** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| forwarding | tosca.capabilities.nfv.Forwarding |  | NfpPositionElement forwarding capability |

#### 7.8.5.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.NfpPositionElement  derived\_from: tosca.nodes.Root  description: node definition of NfpPositionElement  capabilities:  forwarding:  type: tosca.capabilities.nfv.Forwarding  requirements:  - profile\_element:  capability: tosca.capabilities.nfv.Forwarding  relationship: tosca.relationships.nfv.ForwardTo  occurrences: [ 1, 2 ] #When the number of occurrences is 1, the ingress and egress traffic is associated to a single VnfExtCp or Sap; When the number of occurrences is 2, the ingress VnfExtCp or Sap is associated to the first value and the egress VnfExtCp or Sap is associated to the second value. |

#### 7.8.5.7 Artifact

None.

#### 7.8.5.8 Additional Requirements

The valid node types for the "profile\_element" requirements shall be limited to VNF, PNF, NS and SAP.

#### 7.8.5.9 Example

None.

### 7.8.6 tosca.nodes.nfv.NFP

#### 7.8.6.1 Description

The NFP node type associates traffic flow criteria to a list of descriptors associated to the connection points and service access points to be visited by traffic flows matching these criteria. Table 7.8.6.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.8.6.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NFP |
| **Type Qualified Name** | toscanfv:NFP |
| **Type URI** | tosca.nodes.nfv.NFP |

#### 7.8.6.2 Properties

None.

#### 7.8.6.3 Attributes

None.

#### 7.8.6.4 Requirements

The requirements of the NFP node type shall comply with the provisions set out in table 7.8.6.4-1.

Table 7.8.6.4-1: Requirements

| Name | Required | Capability type | Constraints | Description |
| --- | --- | --- | --- | --- |
| nfp\_position | yes | tosca.capabilities.nfv.Forwarding |  |  |

#### 7.8.6.5 Capabilities

None.

#### 7.8.6.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.NFP:  derived\_from: tosca.nodes.Root  description: node definition of NFP  requirements:  - nfp\_position:  capability: tosca.capabilities.nfv.Forwarding  node: tosca.nodes.nfv.NfpPosition  relationship: tosca.relationships.nfv.ForwardTo  occurrences: [ 1, UNBOUNDED ] |

### 7.8.7 tosca.nodes.nfv.NfpPosition

#### 7.8.7.1 Description

The NfpPosition node type describes the reference of one or more NfpPositionElements and rules on how to route traffic flows among VnfExtCp or SAP instances corresponding to these elements. Table 7.8.7.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

NOTE: The NfpPosition and NfpPositionElement node types of the VNFFG model describe the entities in VIM for enabling packets/frames to traverse the constituent VNFs, PNFs or Nested NSs of the Network Forwarding Path.

Table 7.8.7.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NfpPosition |
| **Type Qualified Name** | toscanfv:NfpPosition |
| **Type URI** | tosca.nodes.nfv.NfpPosition |

#### 7.8.7.2 Properties

The properties of the NfpPosition node type shall comply with the provisions set out in table 7.8.7.2-1.

Table 7.8.7.2-1: Properties

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| forwarding\_behaviour | no | string | Possible values:  "all",  "lb",  "ff" | Identifies a rule to apply to forward traffic to CP or SAP instances corresponding to the referenced NfpPositionElement(s).  The minimum list of rules to be supported shall include:  all = Traffic flows shall be forwarded simultaneously to all CP or SAP instances created from the referenced CP profile(s).  lb (load balancing) = Traffic flows shall be forwarded to one CP or SAP instance created from the referenced CP profile(s) selected based on a load-balancing algorithm.  The following value may be used as well:  ff (fast failover) = Traffic flows shall be forwarded to the next CP or SAP in case they cannot be forwarded to a CP or SAP instance created from the referenced CP profile(s).  See note. |
| forwarding\_behaviour\_input\_parameters | no | map of string |  | Provides input parameters to configure the forwarding behaviour (e.g. identifies a load balancing algorithm). This property is reserved for future use in the present document. |
| NOTE: When no rules are provided and there are multiple CP or SAP instances corresponding to the referenced CP profile(s), the VIM and/or the NFVI are expected to apply NFP-independent rules determined by means outside the scope of the present document. | | | | |

#### 7.8.7.3 Attributes

None.

#### 7.8.7.4 Requirements

The requirements of the NfpPosition node type shall comply with the provisions set out in table 7.8.7.4-1.

Table 7.8.7.4-1: Requirements

| **Name** | **Required** | **Capability type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| element | yes | tosca.capabilities.nfv.Forwarding |  | Specifies that an NfpPosition requires a node that has a forwarding capability. |

#### 7.8.7.5 Capabilities

The capabilities of the NfpPosition node type shall comply with the provisions set out in table 7.8.7.5-1.

Table 7.8.7.5-1: Capabilities

| **Name** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| forwarding | tosca.capabilities.nfv.Forwarding |  | NfpPosition forwarding capability |

#### 7.8.7.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.NfpPosition:  derived\_from: tosca.nodes.Root  description: node definition of NFP position  properties:  forwarding\_behaviour:  type: string  description: Identifies a rule to apply to forward traffic to CP or SAP instances corresponding to the referenced NfpPositionElement(s).  constraints:  - valid\_values: [ all, lb, ff ]  required: false  # forwarding\_behaviour\_input\_parameters:  # description: Provides input parameters to configure the forwarding behaviour.  # type: map  # required: false  # entry\_schema:  # type: string  capabilities:  forwarding:  type: tosca.capabilities.nfv.Forwarding  requirements:  - element:  capability: tosca.capabilities.nfv.Forwarding  node: tosca.nodes.nfv.NfpPositionElement  relationship: tosca.relationships.nfv.ForwardTo  occurrences: [ 1, UNBOUNDED ] |

#### 7.8.7.7 Artifact

None.

#### 7.8.7.8 Additional Requirements

None.

#### 7.8.7.9 Example

See clause A.14.

### 7.8.8 tosca.nodes.nfv.Forwarding

#### 7.8.8.1 Description

The Forwarding node type represents a point in the NS topology that can participate as a forwarding target in a network forwarding path. A template of this type is inserted between a virtual link (VirtualLinkable) requirement of a VNF/PNF node template (in effect, an external connection point of the VNF/PNF) or a Sap and the virtual link template satisfying this requirement. Table 7.8.8.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

NOTE: The Forwarding node type is only used for the VNFFGD design. A node template with this type is only present in an NSD if at least one template of the VNFFG group type is included in the NSD.

Table 7.8.8.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | Forwarding |
| **Type Qualified Name** | toscanfv:Forwarding |
| **Type URI** | tosca.nodes.nfv.Forwarding |

#### 7.8.8.2 Properties

None.

#### 7.8.8.3 Attributes

None.

#### 7.8.8.4 Requirements

The requirements of the Forwarding node type shall comply with the provisions set out in table 7.8.8.4-1.

Table 7.8.8.4-1: Requirements

| **Name** | **Capability Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| virtual\_link | tosca.capabilities.nfv.VirtualLinkable |  | Describes the requirement for linking to a virtual linkable node type. |

#### 7.8.8.5 Capabilities

The capabilities of the Forwarding node type shall comply with the provisions set out in table 7.8.8.5-1.

Table 7.8.8.5-1: Capabilities

| **Name** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| forwarding | tosca.capabilities.nfv.Forwarding |  | The forwarding capability exposed by the node. |
| virtual\_linkable | tosca.capabilities.nfv.VirtualLinkable |  | The virtual linkable capability exposed by the node. |

#### 7.8.5.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.Forwarding:  derived\_from: tosca.nodes.Root  capabilities:  virtual\_linkable:  type: tosca.capabilities.nfv.VirtualLinkable  forwarding:  type: tosca.capabilities.nfv.Forwarding  occurrences: [ 1, 2 ] #When the number of occurrences is 1, the ingress and egress traffic is associated to a single VnfExtCp, PnfExtCp or Sap; When the number of occurrences is 2, the ingress VnfExtCp, PnfExtCp or Sap is associated to the first value and the egress VnfExtCp, PnfExtCp or Sap is associated to the second value.  requirements:  - virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo |

#### 7.8.8.7 Artifact

None.

#### 7.8.8.8 Additional Requirements

None.

#### 7.8.8.9 Example

See clause A.14.

## 7.9 Group Types

### 7.9.1 tosca.groups.nfv.NsPlacementGroup

#### 7.9.1.1 Description

The NsPlacementGroup group type is used for 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 an NSD.

Table 7.9.1.1-1 specifies the declared names for this group type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.9.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsPlacementGroup |
| **Type Qualified Name** | toscanfv:NsPlacementGroup |
| **Type URI** | tosca.groups.nfv.NsPlacementGroup |

#### 7.9.1.2 Properties

The properties of the NsPlacementGroup group type shall comply with the provisions set out in table 7.9.1.2-1.

Table 7.9.1.2-1: Properties

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

#### 7.9.1.3 Definition

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

|  |
| --- |
| tosca.groups.nfv.NsPlacementGroup:  derived\_from: tosca.groups.Root  description: NsPlacementGroup is used for 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.  properties:  description:  type: string  description: Human readable description of the group  required: true  members: [tosca.nodes.nfv.VNF, tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NS] |

#### 7.9.1.4 Additional Requirements

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

### 7.9.2 tosca.groups.nfv.VNFFG

#### 7.9.2.1 Description

The VNF Forwarding Graph (VNFFG) group type describes a topology of the NS or a portion of the NS and optionally forwarding rules, applicable to the traffic conveyed over this topology, as defined by ETSI GS NFV-IFA 014 [2]. Table 7.9.2.1-1 specifies the declared names for this group type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.9.2.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VNFFG |
| **Type Qualified Name** | toscanfv:VNFFG |
| **Type URI** | tosca.groups.nfv.VNFFG |

#### 7.9.2.2 Properties

The properties of the VNFFG group type shall comply with the provisions set out in table 7.9.2.2-1.

Table 7.9.2.2-1: properties

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

#### 7.9.2.3 Definition

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

|  |
| --- |
| tosca.groups.nfv.VNFFG:  derived\_from: tosca.groups.Root  description: the VNFFG group type describes a topology of the NS or a portion of the NS, and optionally forwarding rules, applicable to the traffic conveyed over this topology  properties:  description:  type: string  description: Human readable description of the group  required: true  members:[ tosca.nodes.nfv.NFP, tosca.nodes.nfv.VNF, tosca.nodes.nfv.PNF, tosca.nodes.nfv.NS, tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NfpPositionElement ] |

#### 7.9.2.4 Additional Requirements

None.

#### 7.9.2.5 Example

See clause A.14.

## 7.10 Policy Types

### 7.10.1 NsAffinityRule, NsAntiAffinityRule

#### 7.10.1.1 Description

The NsAffinityRule and NsAntiAffinityRule policy describes the affinity or anti-affinity rules applicable for the defined target.

If there is only one node template with node type tosca.nodes.nfv.VNF or tosca.nodes.nfv.NsVirtualLink set as the targets, the NsAffinityRule or NsAntiAffinityRule 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 NsAffinityRule or NsAntiAffinityRule 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].

Tables 7.10.1.1-1 and 7.10.1.1-2 specify the declared names for the policy types. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.10.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsAffinityRule |
| **Type Qualified Name** | toscanfv:NsAffinityRule |
| **Type URI** | tosca.policies.nfv.NsAffinityRule |

Table 7.10.1.1-2: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsAntiAffinityRule |
| **Type Qualified Name** | toscanfv:NsAntiAffinityRule |
| **Type URI** | tosca.policies.nfv.NsAntiAffinityRule |

#### 7.10.1.2 Properties

The properties of the NsAffinityRule and NsAntiAffinityRule types shall comply with the provisions set out in table 7.10.1.2-1.

Table 7.10.1.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| scope | Yes | String | Valid values : "nfvi\_pop", "zone", "zone\_group", "nfvi\_node". | Specifies the scope of the local affinity rule. |

#### 7.10.1.3 Targets

The targets of the NsAffinityRule and NsAntiAffinityRule policy types shall comply with the provisions set out in table 7.10.1.3-1 when used in an NSD.

Table 7.10.1.3-1: Targets

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| targets | Yes | string[] | Valid types:  tosca.nodes.nfv.VNF,  tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NS,  tosca.groups.nfv.NsPlacementGroup | In case of LocalAffinityOrAntiAffinityRule as defined in ETSI GS NFV-IFA 014 [2], the valid type of the targets is tosca.nodes.nfv.VNF or tosca.nodes.nfv.NsVirtualLink.  In case of affinityOrAntiAffinityGroup as defined in ETSI GS NFV-IFA 014 [2], the valid types of the targets are: tosca.nodes.nfv.VNF, tosca.nodes.nfv.NsVirtualLink and tosca.nodes.nfv.NS or a tosca.groups.nfv.NsPlacementGroup. |

#### 7.10.1.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.NsAffinityRule:  derived\_from: tosca.policies.Placement  description: The NsAffinityRule describes the affinity rules applicable for the defined targets  properties:  scope:  type: string  description: Specifies the scope of the local affinity rule.  required: true  constraints:  - valid\_values: [ nfvi\_node, zone, zone\_group, nfvi\_pop ]  targets: [tosca.nodes.nfv.VNF, tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NS, tosca.groups.nfv.NsPlacementGroup ] |

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

|  |
| --- |
| tosca.policies.nfv.NsAntiAffinityRule:  derived\_from: tosca.policies.Placement  description: The NsAntiAffinityRule describes the anti-affinity rules applicable for the defined targets  properties:  scope:  type: string  description: Specifies the scope of the local affinity rule..  required: true  constraints:  - valid\_values: [ nfvi\_node, zone, zone\_group, nfvi\_pop ]  targets: [tosca.nodes.nfv.VNF, tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NS, tosca.groups.nfv.NsPlacementGroup ] |

#### 7.10.1.5 Examples

The following example template fragments illustrate the concepts:

|  |
| --- |
| node\_templates:  VNF\_1:  type: tosca.nodes.nfv.exampleVNF  policies:  policy\_affinity\_local\_VNF\_1:  type: tosca.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: tosca.nodes.nfv.Vdu.exampleVNF\_1  VNF\_2:  type: tosca.nodes.nfv.Vdu.exampleVNF\_2  groups:  affinityOrAntiAffinityGroup\_1:  type: tosca.groups.nfv.PlacementGroup  members: [ VNF\_1, VNF\_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 VNF\_1 and VNF\_2 as members.

### 7.10.2 tosca.policies.nfv.NsSecurityGroupRule

#### 7.10.2.1 Description

The NsSecurityGroupRule policy type when used in an NSD specifies the matching criteria for the ingress and/or egress traffic to and from visited SAPs. Table 7.10.2.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.10.2.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsSecurityGroupRule |
| **Type Qualified Name** | toscanfv:NsSecurityGroupRule |
| **Type URI** | tosca.policies.nfv.NsSecurityGroupRule |

#### 7.10.2.2 Properties

None

#### 7.10.2.3 targets

The targets of the SecurityGroupRule policy types shall comply with the provisions set out in table 7.10.2.3-1.

Table 7.10.2.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string[] | Valid types:  tosca.nodes.nfv.Sap. | Target connection points of Sap. |

#### 7.10.2.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.NsSecurityGroupRule:  derived\_from: tosca.policies.nfv.Abstract.SecurityGroupRule  description: The NsSecurityGroupRule type is a policy type specified the matching criteria for the ingress and/or egress traffic to/from visited SAPs.  targets: [ tosca.nodes.nfv.Sap ] |

#### 7.10.2.5 Additional Requirements

None.

### 7.10.3 tosca.policies.nfv.NfpRule

#### 7.10.3.1 Description

The NfpRule policy type represents the NFP rule attribute of the Nfpd information element as defined in ETSI GS NFV-IFA 014 [2], which describes the conditions that shall be met in order for the NFP to be applicable to the packet. Table 7.10.3.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.10.3.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NfpRule |
| **Type Qualified Name** | toscanfv:NfpRule |
| **Type URI** | tosca.policies.nfv.NfpRule |

#### 7.10.3.2 Properties

The properties of the NFP policy type shall comply with the provisions set out in table 7.10.3.2-1.

Table 7.10.3.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| ether\_destination\_address | no | string |  | Indicates a destination Mac address. |
| ether\_source\_address | no | string |  | Indicates a source Mac address. |
| ether\_type | no | string | ipv4, ipv6 | Indicates the protocol carried over the Ethernet layer. |
| vlan\_tag | no | list of string |  | Indicates a VLAN identifier in an IEEE 802.1Q-2014 tag [16].  Multiple tags can be included for QinQ stacking. |
| protocol | no | string |  | Indicates the L4 protocol, For IPv4 [17] this corresponds to the field called "Protocol" to identify the next level protocol. For IPv6 [18] this corresponds to the field called the "Next Header" field.  Permitted values: Any keyword defined in the IANA [19] protocol registry, e.g.:   * TCP * UDP * ICMP |
| dscp | no | string |  | For IPv4 [17] a string of "0" and "1" digits that corresponds to the 6-bit Differentiated Services Code Point (DSCP) field of the IP header.  For IPv6 [18] a string of "0" and "1" digits that corresponds to the 6 differentiated services bits of the traffic class header field. |
| source\_port\_range | no | range | 0 - 65535 | Indicates a range of source ports. |
| destination\_port\_range | no | range | 0 - 65535 | Indicates a range of destination ports. |
| source\_ip\_address\_prefix | no | string |  | Indicates the source IP address range in CIDR format. |
| destination\_ip\_address\_prefix | no | string |  | Indicates the destination IP address range in CIDR format. |
| extended\_criteria | no | list of tosca.datatypes.nfv.Mask |  | Indicates values of specific bits in a frame. |

#### 7.10.3.3 Targets

The targets of the NfpRule policy types shall comply with the provisions set out in table 7.10.3.3-1.

Table 7.10.3.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string[] | Valid types:  tosca.nodes.nfv.NFP. | The NFPs to which the rule applies. |

#### 7.10.3.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.NfpRule:  derived\_from: tosca.policies.Root  description: policy definition of NfpRule  properties:  ether\_destination\_address:  description: Indicates a destination Mac address.  type: string  required: false  ether\_source\_address:  description: Indicates a source Mac address.  type: string  required: false  ether\_type:  description: Indicates the protocol carried over the Ethernet layer.  type: string  constraints:  - valid\_values: [ ipv4, ipv6 ]  required: false  vlan\_tag:  description: Indicates a VLAN identifier in an IEEE 802.1Q-2014 tag [16]. Multiple tags can be included for QinQ stacking.  type: list  entry\_schema:  type: string  required: false  protocol:  description: 'Indicates the L4 protocol, For IPv4 [17] this corresponds to the field called "Protocol" to identify the next level protocol. For IPv6 [18] this corresponds to the field is called the "Next Header" field. Permitted values: Any keyword defined in the IANA [19] protocol registry.'  type: string  required: false  dscp:  description: For IPv4 [17] a string of "0" and "1" digits that corresponds to the 6-bit Differentiated Services Code Point (DSCP) field of the IP header. For IPv6 [18] a string of "0" and "1" digits that corresponds to the 6 differentiated services bits of the traffic class header field.  type: string  required: false  source\_port\_range:  description: Indicates a range of source ports.  type: range  required: false  constraints:  - in\_range: [ 0, 65535 ]  destination\_port\_range:  description: Indicates a range of destination ports.  type: range  required: false  constraints:  - in\_range: [ 0, 65535 ]  source\_ip\_address\_prefix:  description: Indicates the source IP address range in CIDR format.  type: string  required: false  destination\_ip\_address\_prefix:  description: Indicates the destination IP address range in CIDR format.  type: string  required: false  extended\_criteria:  description: Indicates values of specific bits in a frame.  type: list  entry\_schema:  type: tosca.datatypes.nfv.Mask  required: false  targets: [ tosca.nodes.nfv.NFP ] |

#### 7.10.3.5 Example

None.

### 7.10.4 tosca.policies.nfv.NsMonitoring

#### 7.10.4.1 Description

The NsMonitoring policy type is a policy type representing the virtualised resource related performance metrics to be monitored during the lifetime of network service instance as defined in ETSI GS NFV-IFA 014 [2] and ETSI GS NFV‑IFA 027 [7]. Table 7.10.4.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.10.4.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | NsMonitoring |
| **Type Qualified Name** | toscanfv:NsMonitoring |
| **Type URI** | tosca.policies.nfv.NsMonitoring |

#### 7.10.4.2 Properties

The properties of the NsMonitoring policy type shall comply with the provisions set out in table 7.10.4.2-1.

Table 7.10.4.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| ns\_monitoring\_parameters | yes | list of tosca.datatypes.nfv.NsMonitoringParameter |  | Specifies a virtualised resource related performance metric to be monitored on the NS level. |

#### 7.10.4.3 targets

The targets of the NsMonitoring policy types shall comply with the provisions set out in table 7.10.4.3-1.

Table 7.10.4.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string | Valid types:  tosca.nodes.nfv.NS | Specifies the services node type(s) to which the monitoring policy applies. |

#### 7.10.4.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.NsMonitoring:  derived\_from: tosca.policies.Root  description: Policy type is used to identify information to be monitored during the lifetime of a network service instance as defined in ETSI GS NFV‑IFA 014 [2].  properties:  ns\_monitoring\_parameters:  type: list  description: Specifies a virtualised resource related performance metric to be monitored on the NS level.  required: true  entry\_schema:  type: tosca.datatypes.nfv.NsMonitoringParameter  constraints:  - min\_length: 1  targets: [ tosca.nodes.nfv.NS ] |

#### 7.10.4.5 Additional Requirements

When a policy of this type is specified in an NS service template, the targets set shall only include NS node template names that correspond to this NS or to a nested NS.

### 7.10.5 tosca.policies.nfv.VnfMonitoring

#### 7.10.5.1 Description

The VnfMonitoring policy type is a policy type representing the virtualised resource related performance metrics to be monitored during the lifetime of VNF instance as defined in ETSI GS NFV-IFA 014 [2] and ETSI GS NFV‑IFA 027 [7]. Table 7.10.5.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 7.10.5.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VnfMonitoring |
| **Type Qualified Name** | toscanfv:VnfMonitoring |
| **Type URI** | tosca.policies.nfv.VnfMonitoring |

#### 7.10.5.2 Properties

The properties of the VnfMonitoring policy type shall comply with the provisions set out in table 7.10.5.2-1.

Table 7.10.5.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| vnf\_monitoring\_parameters | yes | list of tosca.datatypes.nfv.VnfMonitoringParameter |  | Specifies a virtualised resource related performance metric to be monitored on the VNF level. |

#### 7.10.5.3 targets

The targets of the VnfMonitoring policy types shall comply with the provisions set out in table 7.10.5.3-1.

Table 7.10.5.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string | Valid types:  tosca.nodes.nfv.VNF | Specifies the VNF node type(s) to which the monitoring policy applies. |

#### 7.10.5.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.VnfMonitoring:  derived\_from: tosca.policies.Root  description: Policy type is used to identify information to be monitored during the lifetime of a VNF instance as defined in ETSI GS NFV-IFA 014 [2].  properties:  vnf\_monitoring\_parameters:  type: list  description: Specifies a virtualised resource related performance metric to be monitored on the NS level.  required: true  entry\_schema:  type: tosca.datatypes.nfv.VnfMonitoringParameter  constraints:  - min\_length: 1  targets: [ tosca.nodes.nfv.VNF ] |

#### 7.10.5.5 Additional Requirements

When a policy of this type is specified in an NS service template, the targets set shall only include VNF node template names representing constituent VNFs for the NS deployment flavour corresponding to this NS.

### 7.10.6 tosca.policies.nfv.Abstract.SecurityGroupRule

#### 7.10.6.1 Description

The Abstract.SecurityGroupRule policy type is defined in clause 9.10.1 of the present document.

## 7.11 NSD TOSCA service template design

### 7.11.1 General

The TOSCA service template design for an NSD in the general case uses two levels of service templates as described in clause 7.11.2. In this design, the top level contains an abstract NS 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 NS has one or multiple deployment flavours.

In the particular case of an NS with only one deployment flavour there is an alternative design which is described in clause 7.11.3 and which uses only one service template.

### 7.11.2 Single or multiple deployment flavour design with TOSCA-Simple-Profile-YAML-v1.2

NSD 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. A separate YAML file with an NS specific node type definition which shall be derived from tosca.nodes.nfv.NS node type as defined in clause 7.8.1 shall be provided and is also considered as a part of an NSD. The top level service template shall be the main entry point of the NSD file structure as specified in ETSI GS NFV-SOL 007 [i.11], i.e. the Entry-definitions file.

See clause A.11 for an example of NSD design with multiple deployment flavours.

The top level service template shall comply with TOSCA-Simple-Profile-YAML-v1.2 [3] and shall include:

1. an import statement referencing the TOSCA types definition file as defined in clause B.3;
2. an import statement referencing the definitions of all low level service template(s);
3. an import statement referencing a yaml file which contains an NS specific node type definition;
4. a topology template with a node template of the NS-specific node type, which shall include the flavour\_id and other properties and the requirements as defined in clause 7.8.1; and
5. optionally, import statements referencing additional NS-specific files containing only type definitions used by this TOSCA service template.

The deployment and lifecycle management of instances of this NS 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:

1. an import statement referencing the TOSCA types definition file as defined in clause B.3;
2. an import statement referencing a yaml file which contains an NS specific node type definition which shall be derived from tosca.nodes.nfv.NS node type as defined in clause 7.8.1;
3. one of more import statements respectively referencing the yaml file which contains the included VNF specific node type definition if any or the included PNF specific node type definition if any or the included NS specific node type definition if any;

If the imported files contain a topology template, this topology template shall be ignored during the parsing of the NSD.

1. optionally, import statements referencing additional NS-specific files containing type definitions used by this TOSCA service template; and
2. a topology template describing the internal topology of the NS with:

* substitution\_mappings indicating:
* the same node type as defined in the NS specific node type definition service template;
* a flavour\_id property value which identifies the DF corresponding to this low level template within the NSD;
* the mapping of the virtual\_link requirements on SAPs;
* a node template referencing the NS specific node type, implementations of the operations of the LCM interface to be executed by the NFVO, if applicable; and
* additional node templates of type VNF, PNF, NS, NsVirtualLink, Sap, etc. that define the topology and composition of the NS flavour, the dependency requirements as defined in TOSCA-Simple-Profile-YAML-v1.2 [3] may be used between different VNF node templates, or between a VNF node template and a nested NS node template, or between different nested NS node templates to specify the order in which instances of the VNFs and/or nested NSs have to be created.

NOTE 1: The format and structure of an NSD file structure is defined in ETSI GS NFV-SOL 007 [i.11].

NOTE 2: All the imported type definition files as indicated either in the top level service template or in any of the lower level service template are considered as parts of an NSD.

When the flavour\_id of an NS has been chosen (e.g. through an input parameter of an NS instantiation request received by a NFVO) among the values included in the NS 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 NSD file structure by using abstract node template matching with substitution as described in TOSCA-Simple-Profile-YAML-v1.2 [3].

### 7.11.3 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 NSD with a single TOSCA service template.

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:

1. an import statement referencing the TOSCA types definition file as defined in clause B.3;
2. one of more import statements respectively referencing the yaml file which contains the included VNF specific node type definition if any or the included PNF specific node type definition if any or the included NS specific node type definition if any;

If the imported files contain a topology template, this topology template shall be ignored during the parsing of the NSD.

1. optionally, import statements referencing additional NS-specific files containing only type definitions used by this TOSCA service template;
2. either an NS specific node type definition derived from the tosca.nodes.nfv.NS node type, as defined in clause 7.8.1 or an import statement referencing a file that contains such definition; and
3. a topology template describing the internal topology of the NS with:

* substitution\_mappings indicating the same NS specific node type and the mapping of the virtual\_link requirements on SAPs;
* a node template of this NS specific 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 NFVO; and
* additional node templates of type VNF, PNF, NS, NsVirtualLink, Sap, etc. that define the topology and composition of the NS flavour, the dependency requirements as defined in TOSCA-Simple-Profile-YAML-v1.2 [3] may be used between different VNF node templates, or between a VNF node template and a nested NS node template, or between different nested NS node templates to specify the order in which instances of the VNFs and/or nested NSs have to be created.

See clause A.8 for an example of NSD 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 NS specific node type, which may appear in an NSD.

NOTE 3: All the imported type definition files as indicated in the service template are considered as parts of an NSD.

# 8 PNFD TOSCA model

## 8.1 Introduction

The PNFD 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 PNFD.

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 PNFD.

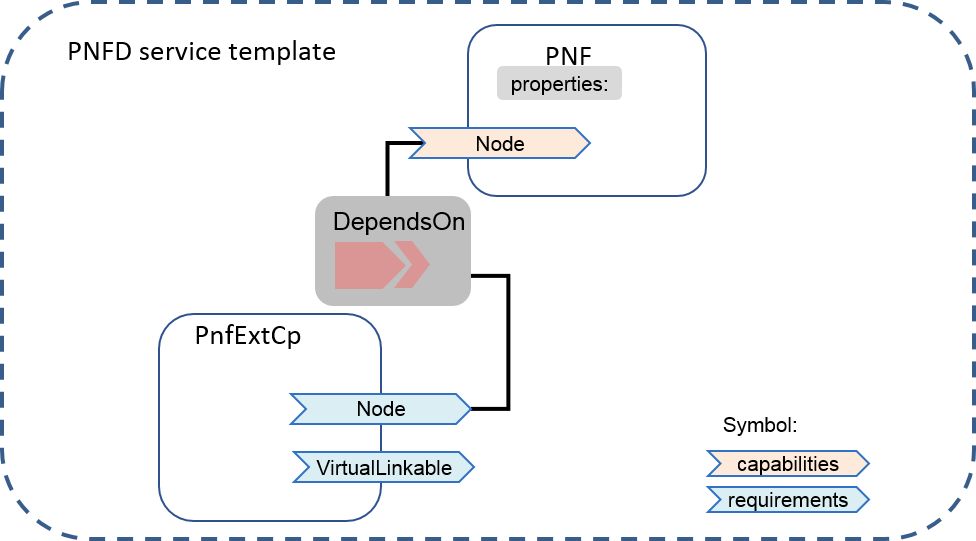


Figure 8.1-1: Service template PNFD overview

## 8.2 Data Types

### 8.2.1 tosca.datatypes.nfv.CpProtocolData

#### 8.2.1.1 Description

The CpProtocolData data type is defined in clause 9.2.6 of the present document.

### 8.2.2 tosca.datatypes.nfv.AddressData

#### 8.2.2.1 Description

The AddressData data type is defined in clause 9.2.3 of the present document.

### 8.2.3 tosca.datatypes.nfv.L2AddressData

#### 8.2.3.1 Description

The L2AddressData data type is defined in clause 9.2.1 of the present document.

### 8.2.4 tosca.datatypes.nfv.L3AddressData

#### 8.2.4.1 Description

The L3AddressData data type is defined in clause 9.2.2 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** | LocationInfo |
| **Type Qualified Name** | toscanfv:LocationInfo |
| **Type URI** | tosca.datatypes.nfv.LocationInfo |

#### 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** |
| --- | --- | --- | --- | --- |
| country\_code | yes | string |  | Shall be a two-letter ISO 3166 [10] country code in capital letters. |
| civic\_address\_element | no | list of tosca.datatypes.nfv.CivicAddressElement |  | Elements composing the civic address where the PNF is deployed. |
| geographic\_coordinates | no | tosca.datatypes.nfv.GeographicCoordinates |  | Geographic coordinates (e.g. Altitude, Longitude, Latitude) 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: tosca.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: tosca.datatypes.nfv.CivicAddressElement  description: Elements composing the civic address where the PNF is deployed.  required: false  geographic\_coordinates:  type: tosca.datatypes.nfv.GeographicCoordinates  description: Geographic coordinates (e.g. Altitude, Longitude, Latitude) 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 tosca.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: tosca.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.

### 8.2.7 tosca.datatypes.nfv.GeographicCoordinates

#### 8.2.7.1 Description

The GeographicCoordinates data type represents a geographic coordinate location as specified in IETF RFC 6225 [21]. Table 8.2.7.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.2.7.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | GeographicCoordinates |
| **Type Qualified Name** | toscanfv: GeographicCoordinates |
| **Type URI** | tosca.datatypes.nfv.GeographicCoordinates |

#### 8.2.7.2 Properties

The properties of the GeographicCoordinates data type shall comply with the provisions set out in table 8.2.7.2-1.

Table 8.2.7.2-1: Properties

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| latitude\_uncertainty | no | string |  | Describe the content of latitude\_uncertainty. The value of latitude\_uncertainty shall comply with LatUnc in section 2.3 of IETF RFC 6225 [21]. |
| latitude | yes | string |  | Describe the content of latitude. The value of latitude shall comply with Latitude in section 2.3 of IETF RFC 6225 [21]. |
| longitude\_ uncertainty | no | string |  | Describe the content of longitude\_uncertainty. The value of longitude\_uncertainty shall comply with LongUnc in section 2.3 of IETF RFC 6225 [21]. |
| longitude | yes | string |  | Describe the content type of longitude. The value of longitude shall comply with Longitude in section 2.3 of IETF RFC 6225 [21]. |
| altitude\_type | yes | string |  | Describe the content type of altitude\_type. The value of altitude\_type shall comply with AType in section 2.4 of IETF RFC 6225 [21]. |
| altitude\_uncertainty | no | string |  | Describe the content of altitude\_uncertainty. The value of altitude\_uncertainty shall comply with AltUnc in section 2.4 of IETF RFC 6225 [21]. |
| altitude | yes | string |  | Describe the content of altitude. The value of altitude shall comply with Altitude in section 2.4 of IETF RFC 6225 [21]. |

#### 8.2.7.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.GeographicCoordinates:  derived\_from: tosca.datatypes.Root  description: Represents an element of a geographic coordinate location as specified in IETF RFC 6225 [21].  properties:  latitude\_uncertainty:  type: string # RFC 6225  description: LatUnc as per RFC 6225  required: false  latitude:  type: string # RFC 6225  description: Latitude value as per RFC 6225  required: true  longitude\_uncertainty:  type: string # RFC 6225  description: LongUnc as per RFC 6225  required: false  longitude:  type: string # RFC 6225  description: Longitude value as per RFC 6225  required: true  altitude\_type:  type: string # RFC 6225  description: AType value as per RFC 6225  required: true  altitude\_uncertainty:  type: string # RFC 6225  description: AltUnc as per RFC 6225  required: false  altitude:  type: string # RFC 6225  description: Altitude value as per RFC 6225  required: true |

#### 8.2.7.4 Examples

None.

#### 8.2.7.5 Additional Requirements

None.

## 8.3 Artifact Types

None.

## 8.4 Capabilities Types

### 8.4.1 tosca.capabilities.nfv.VirtualLinkable

#### 8.4.1.1 Description

The VirtualLinkable capability type is defined in clause 9.4.1 of the present document.

## 8.5 Requirements Types

None.

## 8.6 Relationship Types

### 8.6.1 tosca.relationships.nfv.VirtualLinksTo

#### 8.6.1.1 Description

The VirtualLinksTo relationship type is defined in clause 9.6.1 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. See note. |
| 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.  See note. |
| name | yes | string |  | Name to identify the PNFD. |
| geographical\_location\_info | no | tosca.datatype.nfv.LocationInfo |  | Provides information about the geographical location (e.g. geographic coordinates or address of the building, etc.) of the PNF. |
| NOTE: 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 IETF RFC 4122 [9]. | | | | |

#### 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 | Capability type | Constraints | Description |
| --- | --- | --- | --- | --- |
| virtual\_link | no | tosca.capabilities.nfv.VirtualLinkable |  | Describes the requirements for linking to virtual link |

#### 8.8.1.5 Capabilities

None.

#### 8.8.1.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.PNF:  derived\_from: tosca.nodes.Root  properties:  descriptor\_id: # instead of pnfd\_id  type: string # UUID  required: true  description: Identifier of this PNFD information element. It uniquely identifies the PNFD.  function\_description:  type: string  required: true  description: Describes the PNF function.  provider:  type: string  required: true  description: Identifies the provider of the PNFD.  version:  type: string  required: true  description: Identifies the version of the PNFD.  descriptor\_invariant\_id: # instead of pnfd-invariant-id  type: string # UUID  required: true  description: Identifier of this PNFD in a version independent manner. This attribute is invariant across versions of PNFD.  name:  type: string  required: true  description: Name to identify the PNFD.  geographical\_location\_info:  type: tosca.datatypes.nfv.LocationInfo  required: false  description: Provides information about the geographical location (e.g. geographic coordinates or address of the building, etc.) of the PNF.  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:

1. The node type shall be derived from: tosca.nodes.nfv.PNF.
2. 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.
3. The requirements of tosca.nodes.nfv.PNF shall be preserved.
4. 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 PNFD 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. In the latter case, the virtual\_link requirement should be included in the node type definition with occurrences [ 0, 0 ].

#### 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** | PnfExtCp |
| **Type Qualified Name** | toscanfv:PnfExtCp |
| **Type URI** | tosca.nodes.nfv.PnfExtCp |

#### 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 | Capability 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. |

#### 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: tosca.nodes.nfv.Cp  description: node definition of PnfExtCp.  requirements:  - external\_virtual\_link:  capability: tosca.capabilities.nfv.VirtualLinkable  relationship: tosca.relationships.nfv.VirtualLinksTo  occurrences: [0, 1] |

### 8.8.3 tosca.nodes.nfv.Cp

#### 8.8.3.1 Description

The Cp node type is defined in clause 9.8.1 of the present document.

## 8.9 Group Types

None.

## 8.10 Policy Types

### 8.10.1 tosca.policies.nfv.PnfSecurityGroupRule

#### 8.10.1.1 Description

The PnfSecurityGroupRule policy type when used in a PNFD specifies the matching criteria for the ingress and/or egress traffic to and from visited PNF external connection points. Table 8.10.1.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 8.10.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | PnfSecurityGroupRule |
| **Type Qualified Name** | toscanfv:PnfSecurityGroupRule |
| **Type URI** | tosca.policies.nfv.PnfSecurityGroupRule |

#### 8.10.1.2 Properties

None.

#### 8.10.1.3 targets

The targets of the SecurityGroupRule policy types shall comply with the provisions set out in table 8.10.1.3-1.

Table 8.10.1.3-1: Targets

| **Name** | **Required** | **Type** | **Constraints** | **Description** |
| --- | --- | --- | --- | --- |
| targets | yes | string[] | Valid types:  tosca.nodes.nfv.PnfExtCp | Target connection points of PnfExtCp |

#### 8.10.1.4 Definition

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

|  |
| --- |
| tosca.policies.nfv.PnfSecurityGroupRule:  derived\_from: tosca.policies.nfv.Abstract.SecurityGroupRule  description: The PnfSecurityGroupRule type is a policy type specified the matching criteria for the ingress and/or egress traffic to/from visited PNF external connection points.  targets: [ tosca.nodes.nfv.PnfExtCp ] |

#### 8.10.1.5 Additional Requirements

None.

### 8.10.2 tosca.policies.nfv.Abstract.SecurityGroupRule

#### 8.10.2.1 Description

The Abstract.SecurityGroupRule policy type is defined in clause 9.10.1 of the present document.

## 8.11 PNFD TOSCA service template design

### 8.11.1 General

One single TOSCA service template is used to design a PNFD which shall comply with TOSCA-Simple-Profile-YAML-v1.2 [3] and includes:

NOTE 1: The present document does not preclude this pattern using TOSCA-Simple-Profile-YAML-v1.1 [4].

1. an import statement referencing the TOSCA types definition file as defined in clause B.4;
2. a PNF specific node type definition derived from the tosca.nodes.nfv.PNF node type, as defined in clause 8.8.1; and
3. a topology template describing the internal topology of the PNF with:

* substitution\_mappings indicating the same PNF specific node type and the mapping of the virtual\_link requirements on PNF external connection point;
* a node template of this PNF specific node type with the properties as defined in tosca.nodes.nfv.PNF; and
* additional node templates of type PnfExtCp that define the connection information of the PNF.

See clause A.10 for an example of PNFD design.

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 PNF specific node type, which may appear in an NSD.

# 9 Common Definitions

## 9.1 Introduction

This clause defines the TOSCA type definitions which are used by at least two types of deployment templates among those identified in clause 5.1.

## 9.2 Data Types

### 9.2.1 tosca.datatypes.nfv.L2AddressData

#### 9.2.1.1 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 9.2.1.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.1.1-1: Type name, shorthand, and URI

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

#### 9.2.1.2 Properties

The properties of the L2AddressData data type shall comply with the provisions set out in table 9.2.1.2-1.

Table 9.2.1.2-1: 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. |

#### 9.2.1.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.L2AddressData:  derived\_from: tosca.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 |

#### 9.2.1.4 Examples

|  |
| --- |
| <some\_tosca\_entity>:  properties:  l2\_address\_data:  mac\_address\_assignment: true |

#### 9.2.1.5 Additional Requirements

None.

### 9.2.2 tosca.datatypes.nfv.L3AddressData

#### 9.2.2.1 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 9.2.2.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.2.1-1: Type name, shorthand, and URI

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

#### 9.2.2.2 Properties

The properties of the L3AddressData data type shall comply with the provisions set out in table 9.2.2.2-1.

Table 9.2.2.2-1: 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 connection point. |
| number\_of\_ip\_address | no | Integer | greater\_than: 0 | Minimum number of IP addresses to be assigned. |

#### 9.2.2.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.L3AddressData:  derived\_from: tosca.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 connection point  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 |

#### 9.2.2.4 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 |

#### 9.2.2.5 Additional Requirements

None.

### 9.2.3 tosca.datatypes.nfv.AddressData

#### 9.2.3.1 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 9.2.3.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.3.1-1: Type name, shorthand, and URI

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

#### 9.2.3.2 Properties

The properties of the AddressData data type shall comply with the provisions set out in table 9.2.3.2-1.

Table 9.2.3.2-1: 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. |

#### 9.2.3.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.AddressData:  derived\_from: tosca.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: tosca.datatypes.nfv.L2AddressData  description: Provides the information on the MAC addresses to be assigned to a connection point.  required: false  l3\_address\_data:  type: tosca.datatypes.nfv.L3AddressData  description: Provides the information on the IP addresses to be assigned to a connection point  required: false |

#### 9.2.3.4 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 |

#### 9.2.3.5 Additional Requirements

None.

### 9.2.4 tosca.datatypes.nfv.ConnectivityType

#### 9.2.4.1 Description

The ConnectivityType data type describes the protocol exposed by a virtual link and the flow pattern supported by the virtual link, as defined in ETSI GS NFV-IFA 011 [1]. Table 9.2.4.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.4.1-1: Type name, shorthand, and URI

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

#### 9.2.4.2 Properties

The properties of the ConnectivityType shall comply with the provisions set out in table 9.2.4.2-1.

Table 9.2.4.2-1: 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. |

#### 9.2.4.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.ConnectivityType:  derived\_from: tosca.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 ] |

#### 9.2.4.4 Examples

|  |
| --- |
| <some\_tosca\_entity>:  properties:  connectivity\_type:  layer\_protocol:  - ipv4  flow\_pattern: mesh |

#### 9.2.4.5 Additional Requirements

None.

### 9.2.5 tosca.datatypes.nfv.LinkBitrateRequirements

#### 9.2.5.1 Description

The LinkBitrateRequirements data type describes the requirements in terms of bitrate for a virtual link. Table 9.2.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.5.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | LinkBitrateRequirements |
| **Type Qualified Name** | toscanfv:LinkBitrateRequirements |
| **Type URI** | tosca.datatypes.nfv.LinkBitrateRequirements |

#### 9.2.5.2 Properties

The properties of the LinkBitrateRequirements data type shall comply with the provisions set out in table 9.2.5.2-1.

Table 9.2.5.2-1: 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). |

#### 9.2.5.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.LinkBitrateRequirements:  derived\_from: tosca.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 |

#### 9.2.5.4 Examples

None.

#### 9.2.5.5 Additional Requirements

None.

### 9.2.6 tosca.datatypes.nfv.CpProtocolData

#### 9.2.6.1 Description

The CpProtocolData data type describes and associates the protocol layer that a CP uses together with other protocol and connection point information. Table 9.2.6.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.6.1-1: Type name, shorthand, and URI

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

#### 9.2.6.2 Properties

The properties of the CpProtocolData data type shall comply with the provisions set out in table 9.2.6.2-1.

Table 9.2.6.2-1: 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. |

#### 9.2.6.3 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  description: One of the values of the property layer\_protocols of the CP  required: true  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 |

#### 9.2.6.4 Examples

None.

#### 9.2.6.5 Additional Requirements

None.

### 9.2.7 tosca.datatypes.nfv.Qos

#### 9.2.7.1 Description

The QoS describes QoS data type a given VL used in a VNF deployment flavour. Table 9.2.7.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.7.1-1: Type name, shorthand, and URI

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

#### 9.2.7.2 Properties

The properties of the Qos data type shall comply with the provisions set out in table 9.2.7.2-1.

Table 9.2.7.2-1: 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. |

#### 9.2.7.3 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 ] |

#### 9.2.7.4 Examples

None.

#### 9.2.7.5 Additional Requirements

None.

### 9.2.8 tosca.datatypes.nfv.VnfProfile

#### 9.2.8.1 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 9.2.8.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.2.8.1-1: Type name, shorthand, and URI

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

#### 9.2.8.2 Properties

The properties of the VnfProfile data type shall comply with the provisions set out in table 9.2.8.2-1.

Table 9.2.8.2-1: 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. |

#### 9.2.8.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfProfile:  derived\_from: tosca.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 |

#### 9.2.8.4 Example

None.

#### 9.2.8.5 Additional Requirements

None.

### 9.2.9 tosca.datatypes.nfv.VnfMonitoringParameter

#### 9.2.9.1 Description

This data type provides information on virtualised resource related performance metrics applicable to VNF. Table 9.2.9.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.2 [3].

Table 9.2.9.1-1: Type name, shorthand, and URI

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

#### 9.2.9.2 Properties

The properties of the VnfMonitoringParameter data type shall comply with the provisions set out in table 9.2.9.2-1.

Table 9.2.9.2-1: 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 a performance metric to be monitored.  Performance metric values shall be set to a measurement name defined in clause 7.2 of ETSI GS NFV-IFA 027 [7], without appending a sub-counter. In this case the VNFM computes these measurements from lower-level metrics collected from the VIM. |
| collection\_period | no | scalar-unit.time |  | Describes the periodicity at which to collect the performance information. |

#### 9.2.9.3 Definition

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

|  |
| --- |
| tosca.datatypes.nfv.VnfMonitoringParameter:  derived\_from: tosca.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 to be monitored, 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 periodicity at which to collect the performance information.  required: false  constraints:  - greater\_than: 0 s |

#### 9.2.9.4 Examples

See clause A.8.

#### 9.2.9.5 Additional Requirements

None.

## 9.3 Artifact Types

None.

## 9.4 Capabilities Types

### 9.4.1 tosca.capabilities.nfv.VirtualLinkable

#### 9.4.1.1 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 association between a VduCpd and an intVirtualLinkDesc and the association between a VnfExtCpd and an intVirtualLinkDesc as specified in ETSI GS NFV-IFA 011 [1] as well as the association represented by the NsVirtualLinkConnectivity information element in ETSI GS NFV-IFA 014 [2]. Table 9.4.1.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.4.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VirtualLinkable |
| **Type Qualified Name** | toscanfv:VirtualLinkable |
| **Type URI** | tosca.capabilities.nfv.VirtualLinkable |

#### 9.4.1.2 Properties

None.

#### 9.4.1.3 Definition

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

|  |
| --- |
| tosca.capabilities.nfv.VirtualLinkable:  derived\_from: tosca.capabilities.Node  description: A node type that includes the VirtualLinkable capability indicates that it can be pointed by tosca.relationships.nfv.VirtualLinksTo relationship type |

### 9.4.2 Void

## 9.5 Requirements Types

None.

## 9.6 Relationship Types

### 9.6.1 tosca.relationships.nfv.VirtualLinksTo

#### 9.6.1.1 Description

This relationship type represents an association between the VduCp and VnfVirtualLink node types or the association between either a VnfExtCp, a PnfExtCp or a Sap and an NsVirtualLink node types. Table 9.6.1.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.6.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VirtualLinksTo |
| **Type Qualified Name** | toscanfv:VirtualLinksTo |
| **Type URI** | tosca.relationships.nfv.VirtualLinksTo |

#### 9.6.1.2 Properties

None.

#### 9.6.1.3 Definition

The syntax of the VirtualLinksTo 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 or the association between either a VnfExtCp, a PnfExtCp or a Sap and an NsVirtualLink node types.  valid\_target\_types: [ tosca.capabilities.nfv.VirtualLinkable ] |

### 9.6.2 Void

### 9.6.3 tosca.relationships.nfv.VipVirtualLinksTo

#### 9.6.3.1 Description

This relationship type represents an association between the VipCp and a VnfVirtualLink node types or between the former and an NsVirtualLink node types. Table 9.6.3.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.6.3.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | VipVirtualLinksTo |
| **Type Qualified Name** | toscanfv:VipVirtualLinksTo |
| **Type URI** | tosca.relationships.nfv.VipVirtualLinksTo |

#### 9.6.3.2 Properties

None.

#### 9.6.3.3 Definition

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

|  |
| --- |
| tosca.relationships.nfv.VipVirtualLinksTo:  derived\_from: tosca.relationships.DependsOn  description: Represents an association relationship between the VipCp and a VnfVirtualLink node types or between the former and a NsVirtualLink node types.  valid\_target\_types: [ tosca.capabilities.nfv.VirtualLinkable ] |

## 9.7 Interface Types

None.

## 9.8 Node Types

### 9.8.1 tosca.nodes.nfv.Cp

#### 9.8.1.1 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 9.8.1.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.2 [3].

Table 9.8.1.1-1: Type name, shorthand, and URI

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

#### 9.8.1.2 Properties

The properties of the Cp node type shall comply with the provisions set out in table 9.8.1.2-1.

Table 9.8.1.2-1: 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 tosca.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". |

#### 9.8.1.3 Attributes

None.

#### 9.8.1.4 Requirements

None.

#### 9.8.1.5 Capabilities

None.

#### 9.8.1.6 Definition

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

|  |
| --- |
| tosca.nodes.nfv.Cp:  derived\_from: tosca.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: 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: tosca.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 |

#### 9.8.1.7 Additional requirements

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

## 9.9 Group Types

None.

## 9.10 Policy Types

### 9.10.1 tosca.policies.nfv.Abstract.SecurityGroupRule

#### 9.10.1.1 Description

The Abstract.SecurityGroupRule type represents an abstract policy type without any target requirements. Table 9.10.1.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA‑Simple-Profile-YAML-v1.2 [3].

Table 9.10.1.1-1: Type name, shorthand, and URI

|  |  |
| --- | --- |
| **Shorthand Name** | Abstract.SecurityGroupRule |
| **Type Qualified Name** | toscanfv:Abstract.SecurityGroupRule |
| **Type URI** | tosca.policies.nfv.Abstract.SecurityGroupRule |

#### 9.10.1.2 Properties

The properties of the Abstract.SecurityGroupRule policy type shall comply with the provisions set out in table 9.10.1.2‑1.

Table 9.10.1.2-1: Properties

| Name | Required | Type | Constraints | Description |
| --- | --- | --- | --- | --- |
| description | no | string |  | Human readable description of the security group rule. |
| direction | yes | 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 | yes | string | ipv4, ipv6 | Indicates the protocol carried over the Ethernet layer. |
| protocol | yes | string | see note | Indicates the protocol carried over the IP layer. Permitted values: any protocol defined in the IANA protocol registry [19], e.g. TCP, UDP, ICMP, etc. |
| port\_range\_min | yes | 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 | yes | 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, 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\_cmtp, 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, 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. | | | | |

#### 9.10.1.3 Definition

The syntax of the Abstract.SecurityGroupRule policy type shall comply with the following definition:

|  |
| --- |
| tosca.policies.nfv.Abstract.SecurityGroupRule:  derived\_from: tosca.policies.Root  description: The Abstract.SecurityGroupRule type represents an abstract policy type without any target requirements  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: true  constraints:  - valid\_values: [ ingress, egress ]  default: ingress  ether\_type:  type: string  description: Indicates the protocol carried over the Ethernet layer.  required: true  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: true  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\_cmtp, 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, 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: true  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: true  constraints:  - greater\_or\_equal: 0  - less\_or\_equal: 65535  default: 65535 |

#### 9.10.1.4 Additional Requirements

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.

Annex A (informative):  
Examples

# A.1 Deployment flavour design mapping

## A.1.1 Introduction

This clause describes the main design principle for VNF/NS deployment flavour and the mapping between VNF deployment flavour elements to TOSCA concept.

## A.1.2 Design principle for VNFD deployment flavour

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.

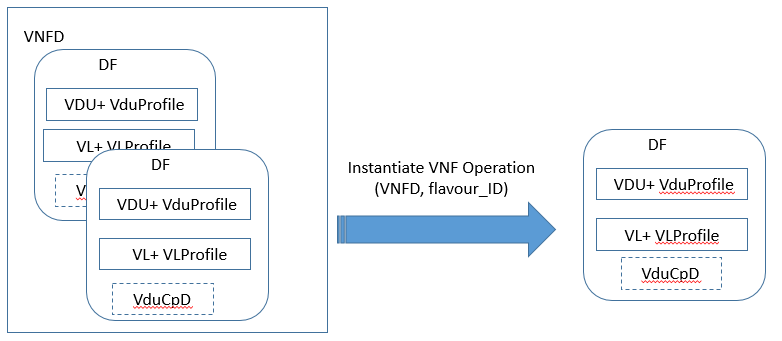


Figure A.1.2-1: General principle for a VNF deployment flavour design

## A.1.3 Design principle for NSD deployment flavour

The design principle for NSD deployment flavour is the same with the VNFD deployment flavour design. Each NS deployment flavour is described as a standalone implementable TOSCA service template, the constituent VNF, virtualLink and nested NS is bound with the corresponding VnfProfile, VirtualLinkProfile and NsProfile 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 NS with the given deployment flavour.

Figure A.1.3-1 shows the general principle for a NS deployment flavour design.

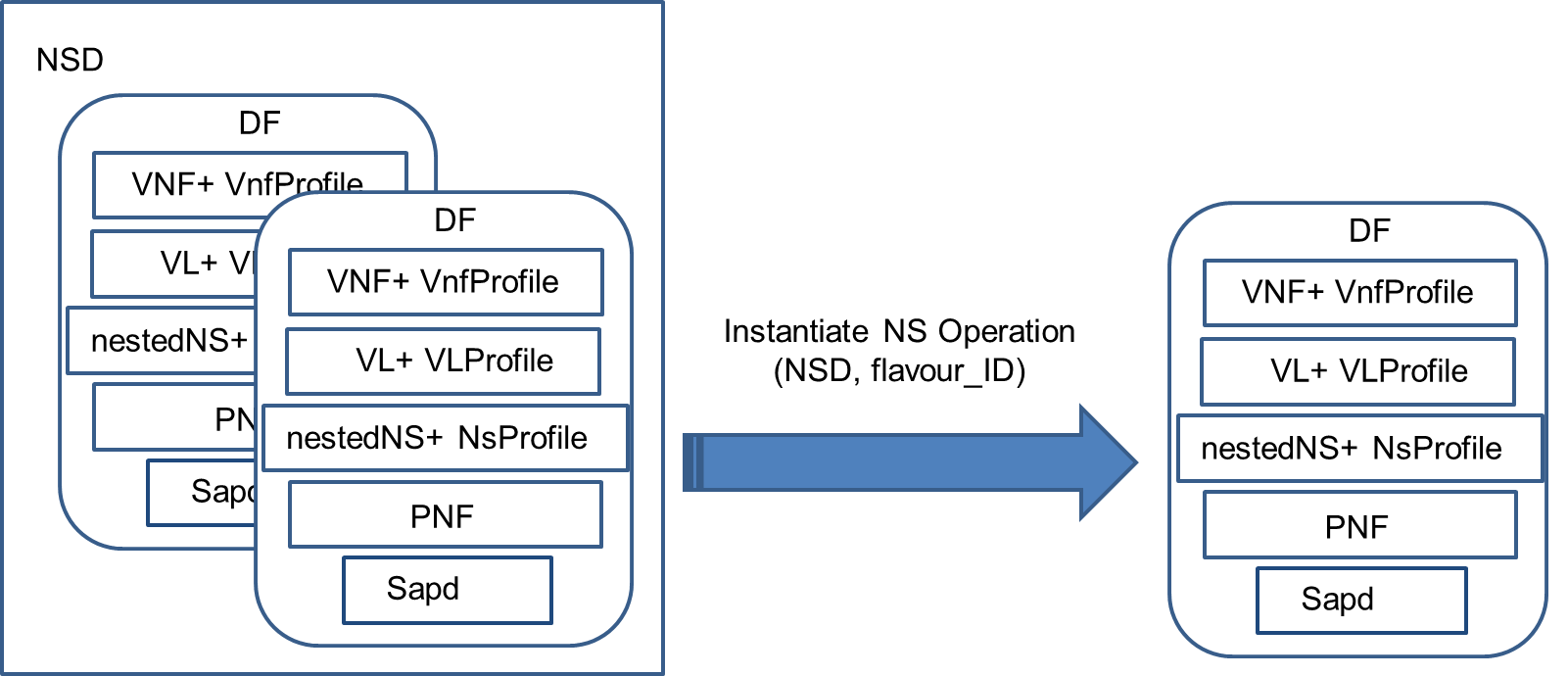


Figure A.1.3-1: General principle for a NS deployment flavour design

# 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.

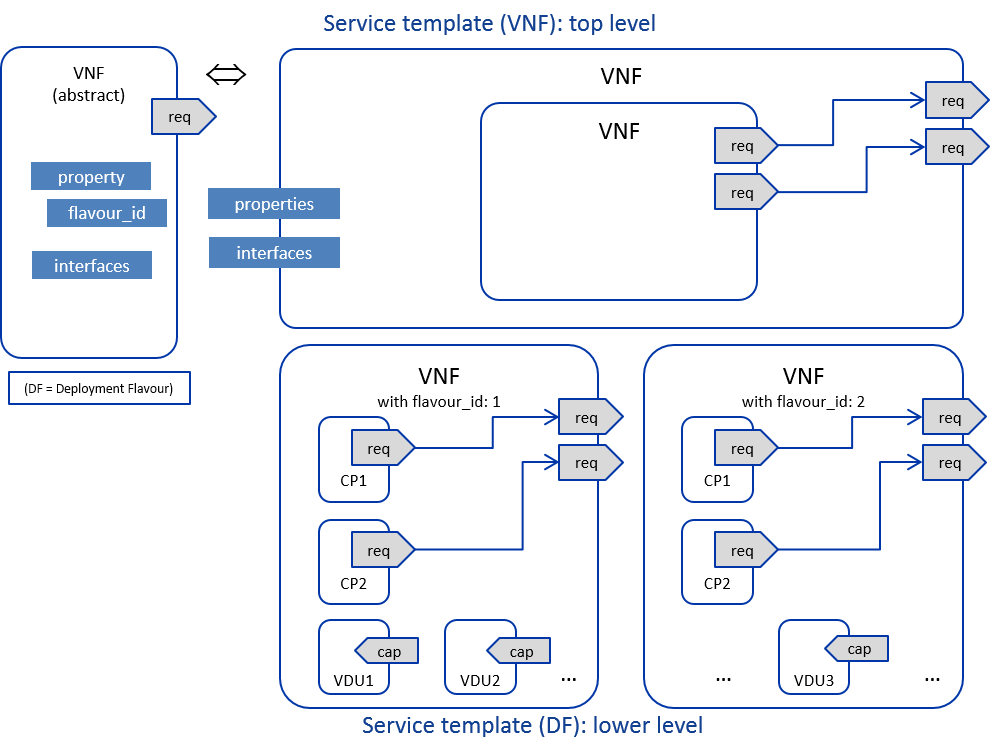


Figure A.2-1: VNFD overview with deployment flavour

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\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

- sunshineVNF.yaml # contains the VNF node type definition

- sunshinedbsimple.vnfd.tosca.yaml

- sunshinedbcomplex.vnfd.tosca.yaml

topology\_template:

inputs:

flavour\_id:

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: flavour\_id }

descriptor\_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

provider: MyCompany

product\_name: SunshineDB

software\_version: '1.0'

descriptor\_version: '1.0'

vnfm\_info:

- '0:MyCompany-1.0.0'

# requirements:

#- virtual\_link\_backend # mapped in lower-level templates

#- virtual\_link\_service # mapped in lower-level templates

# get\_input function would be used to process the template at run time to access the selected flavour id. If the TOSCA functions are not supported by VNFM, the above function may not be needed, what other mechanisms can be used by VNFM which provide the same function are out of scope of the present document.

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\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

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: true

default: value\_2

node\_types:

MyCompany.SunshineDB.1\_0.1\_0:

derived\_from: tosca.nodes.nfv.VNF

properties:

descriptor\_id:

type: string

constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]

default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

provider:

type: string

constraints: [ equal: MyCompany ]

default: MyCompany

product\_name:

type: string

constraints: [ equal: SunshineDB ]

default: SunshineDB

software\_version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

descriptor\_version:

type: string

constraints: [ equal: '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: [ equal: '0:MyCompany-1.0.0' ]

default: [ '0:MyCompany-1.0.0' ]

requirements:

- virtual\_link:

capability: tosca.capabilities.nfv.VirtualLinkable

occurrences: [ 0, 0 ]

- virtual\_link\_backend:

capability: tosca.capabilities.nfv.VirtualLinkable

occurrences: [ 0, 1 ]

- virtual\_link\_service:

capability: tosca.capabilities.nfv.VirtualLinkable

occurrences: [ 0, 1 ]

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 fileis 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".

dbBackend

dbBackendInternalCp

internalVl

dbBackendCp

externalVl

VNF

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\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

- sunshineVNF.yaml # contains the VNF node type definition

node\_types:

MyCompany.nodes.nfv.Vdu.Aux:

derived\_from: tosca.nodes.nfv.Vdu.Compute

properties:

configurable\_properties:

type: MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties

required: false

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: true

default: "MyCustomer"

dns\_server:

type: string

required: true

default: "90.200.250.57"

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: MyCompany.nodes.nfv.Vdu.Aux

properties:

name: dbBackend

description: dbBackend compute node

nfvi\_constraints:

key\_1: value\_1

key\_2: value\_2

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: 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:

algorithm: sha-256

hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d

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

internalVl:

type: tosca.nodes.nfv.VnfVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ ipv4 ]

flow\_pattern: mesh

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".

dbBackend

dbBackendInternalCp

internalVl

serviceNode

serviceNodeInternalCp

serviceNodeCp

dbBackendCp

externalVl

VNF

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\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

- sunshineVNF.yaml # contains the VNF node type definition

node\_types:

MyCompany.nodes.nfv.Vdu.Aux:

derived\_from: tosca.nodes.nfv.Vdu.Compute

properties:

configurable\_properties:

type: MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties

required: false

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: true

default: "MyCustomer"

dns\_server:

type: string

required: true

default: "90.200.250.57"

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: MyCompany.nodes.nfv.Vdu.Aux

properties:

name: dbBackend

description: dbBackend compute node

nfvi\_constraints:

key\_1: value\_1

key\_2: value\_2

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: tosca.nodes.nfv.Vdu.Compute

properties:

name: serviceNode

description: brief description about serviceNode

nfvi\_constraints:

key\_3: value\_3

key\_4: value\_4

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 1

sw\_image\_data:

name: Software of Maria Db

version: '1.0'

checksum:

algorithm: sha-256

hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d

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:

algorithm: sha-256

hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d

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

serviceNodeInternalCp:

type: tosca.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: tosca.nodes.nfv.VnfVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ ipv4 ]

flow\_pattern: mesh

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.

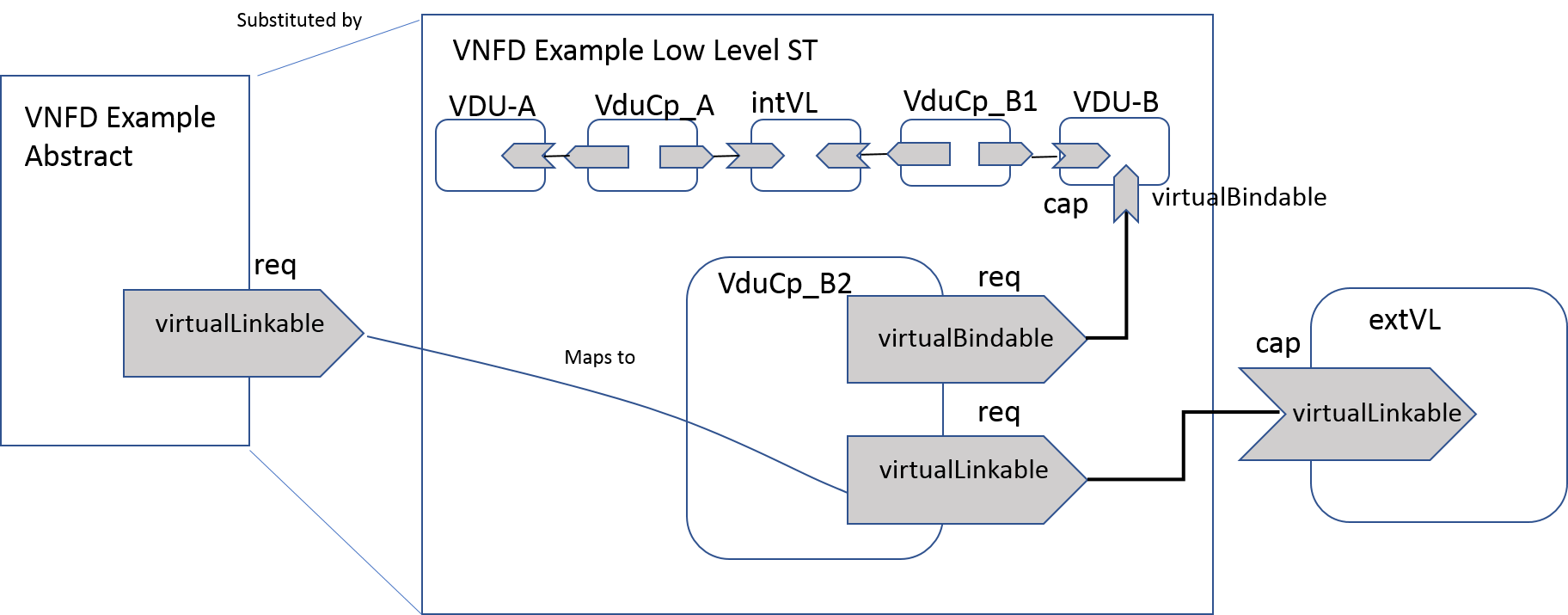


Figure A.3.2-1: VNFD with an VDU connection point acting as VnfExtCp

The following snippet shows the relevant part of the service template:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

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.

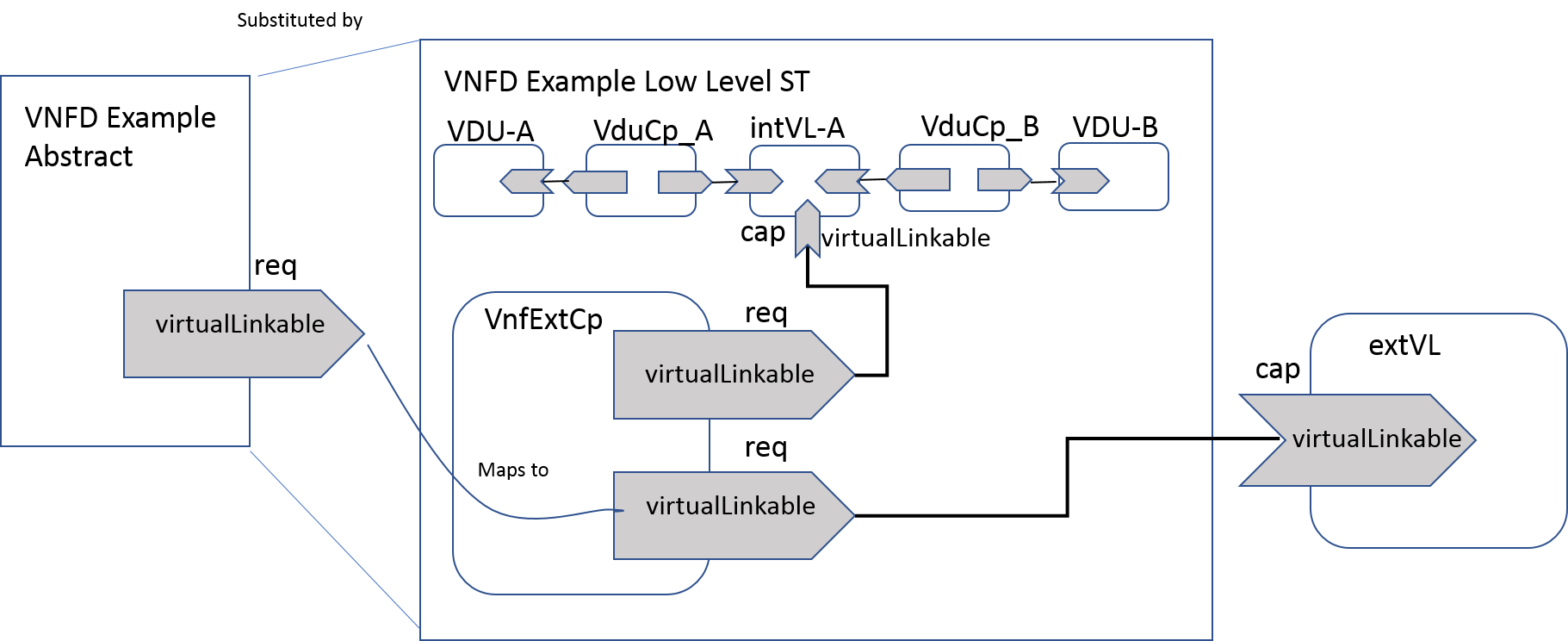


Figure A.3.3-1: VNFD with a VnfExtCp connected to an internal virtual link

The following snippet shows the corresponding node template definition:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

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

# Other node templates e.g. VDU-A, VDU-B, VduCP\_A, etc.:

...

# A.4 VNFD modelling design example by using TOSCA composition

The following example in figure A.4-1 shows a VNF descriptor contains three VDUs, which are interconnected by two virtualLinks.

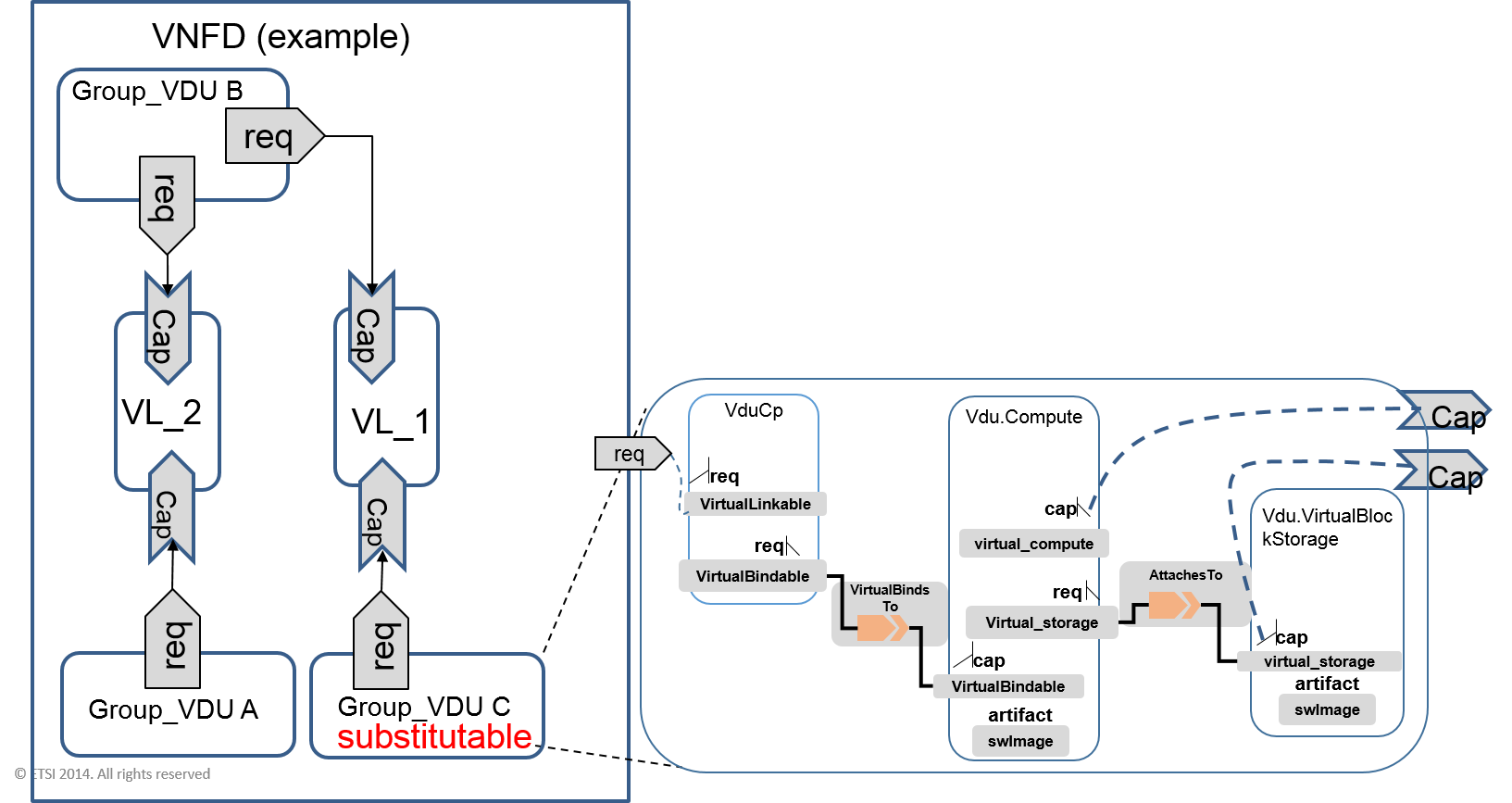


Figure A.4-1: Example of using substitution mapping for the VNFD design

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  imports:  - etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001  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  VL\_2:  type: tosca.nodes.nfv.VnfVirtualLink  # properties omitted here for brevity |

The TOSCA service template example of Group\_VDU C is showing in figure A.4-2.

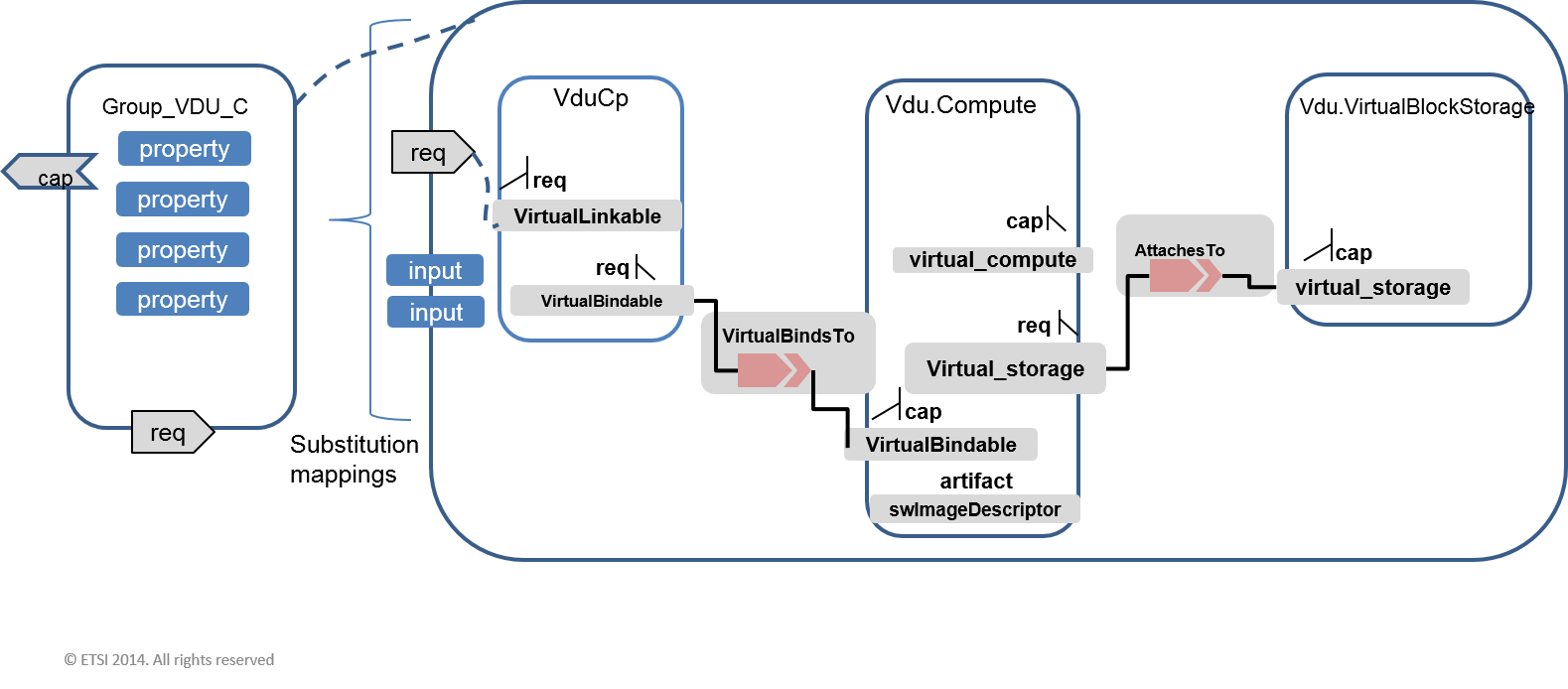


Figure A.4-2: Example of composition Vdu.Compute, Vdu.VirtualBlockStorage and VduCp  
together using TOSCA substitution mapping

|  |
| --- |
| 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:  # omitted here for brevity  artifacts:  sw\_image: # omitted here for brevity  requirements:  - virtual\_storage: vduC\_storage    vduC\_storage:  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage  # properties omitted here for brevity  internalCpd:  type: tosca.nodes.nfv.VduCp  # properties omitted here for brevity  requirements:  # - virtual\_link:  - 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".

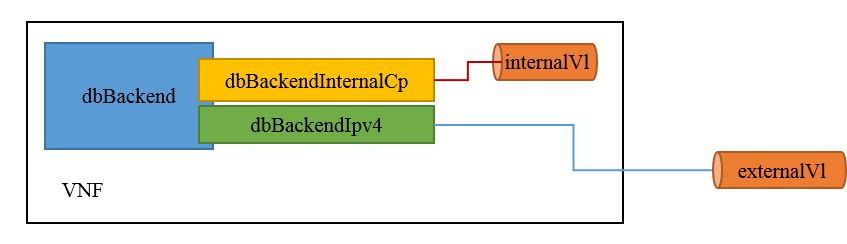


Figure A.5-1: SunshineDB (simple)

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: Relational database, simple

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001 for a VNFD

data\_types:

MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:

derived\_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters

properties:

segmentation\_id\_of\_internalVl:

type: string

required: true

default: 1-4095

parameter\_2:

type: string

required: true

default: value\_2

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: true

default: "MyCustomer"

dns\_server:

type: string

required: true

default: "90.200.250.57"

node\_types:

MyCompany.SunshineDB.1\_0.1\_0:

derived\_from: tosca.nodes.nfv.VNF

properties:

descriptor\_id:

type: string

constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]

default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

provider:

type: string

constraints: [ equal: MyCompany ]

default: MyCompany

product\_name:

type: string

constraints: [ equal: SunshineDB ]

default: SunshineDB

software\_version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

descriptor\_version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

flavour\_id:

type: string

constraints: [ equal: simple ]

default: simple

flavour\_description:

type: string

default: ""

vnfm\_info:

type: list

entry\_schema:

type: string

constraints: [ equal: '0:MyCompany-1.0.0' ]

default: [ '0:MyCompany-1.0.0' ]

interfaces:

Vnflcm:

type: tosca.interfaces.nfv.Vnflcm

instantiate:

inputs:

additional\_parameters:

type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters

#terminate:

MyCompany.nodes.nfv.Vdu.Aux:

derived\_from: tosca.nodes.nfv.Vdu.Compute

properties:

configurable\_properties:

type: MyCompany.datatypes.nfv.AuxVnfcConfigurableProperties

required: false

topology\_template:

substitution\_mappings:

node\_type: MyCompany.SunshineDB.1\_0.1\_0

requirements:

virtual\_link: [ dbBackendIpv4, virtual\_link ] # IPv4 for SQL

inputs:

segmentation\_id\_of\_internalVl:

type: string

required: true

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:

- '0:MyCompany-1.0.0'

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: MyCompany.nodes.nfv.Vdu.Aux

properties:

name: dbbackend

description: dbBackend

nfvi\_constraints:

key\_1: value\_1

key\_2: value\_2

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: tosca.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:

algorithm: sha-256

hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d

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

dbBackendInternalCp:

type: tosca.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: tosca.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

virtual\_link\_protocol\_data:

- associated\_layer\_protocol: ethernet

l2\_protocol\_data:

network\_type: vlan

segmentation\_id: { get\_input: segmentation\_id\_of\_internalVl }

dbBackendIpv4:

type: tosca.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.

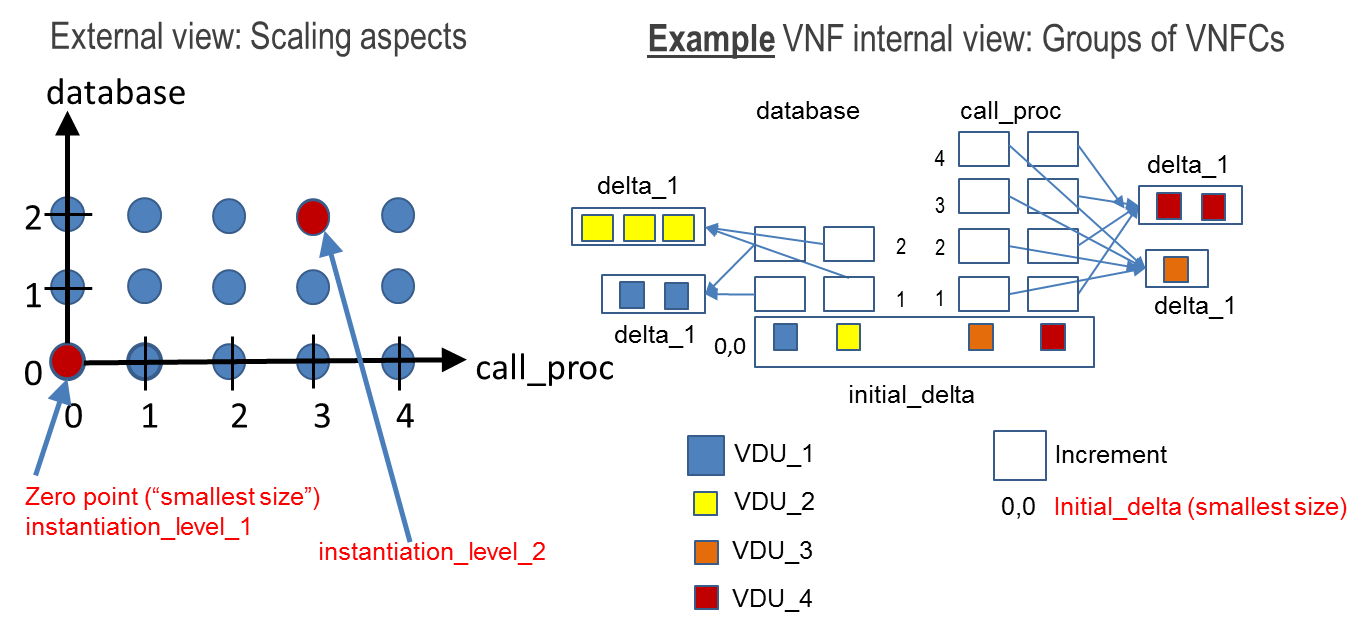


Figure A.6.1-1: Complex scaling example with uniform delta

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: Complex scaling example (uniform delta value) described with policies

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

topology\_template:

node\_templates:

vdu\_1:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 5

nfvi\_constraints:

- ..

vdu\_2:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 7

nfvi\_constraints:

..

vdu\_3:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 5

nfvi\_constraints:

..

vdu\_4:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 9

nfvi\_constraints:

..

vl\_1:

type: tosca.nodes.nfv.VnfVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ ipv4 ]

vl\_profile:

min\_bitrate\_requirements:

root: 1000000

max\_bitrate\_requirements:

root: 2000000

vl\_2:

type: tosca.nodes.nfv.VnfVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ ipv4 ]

vl\_profile:

min\_bitrate\_requirements:

root: 1000000

max\_bitrate\_requirements:

root: 4000000

policies:

- scaling\_aspects:

type: tosca.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 InstantationLevels policies 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.

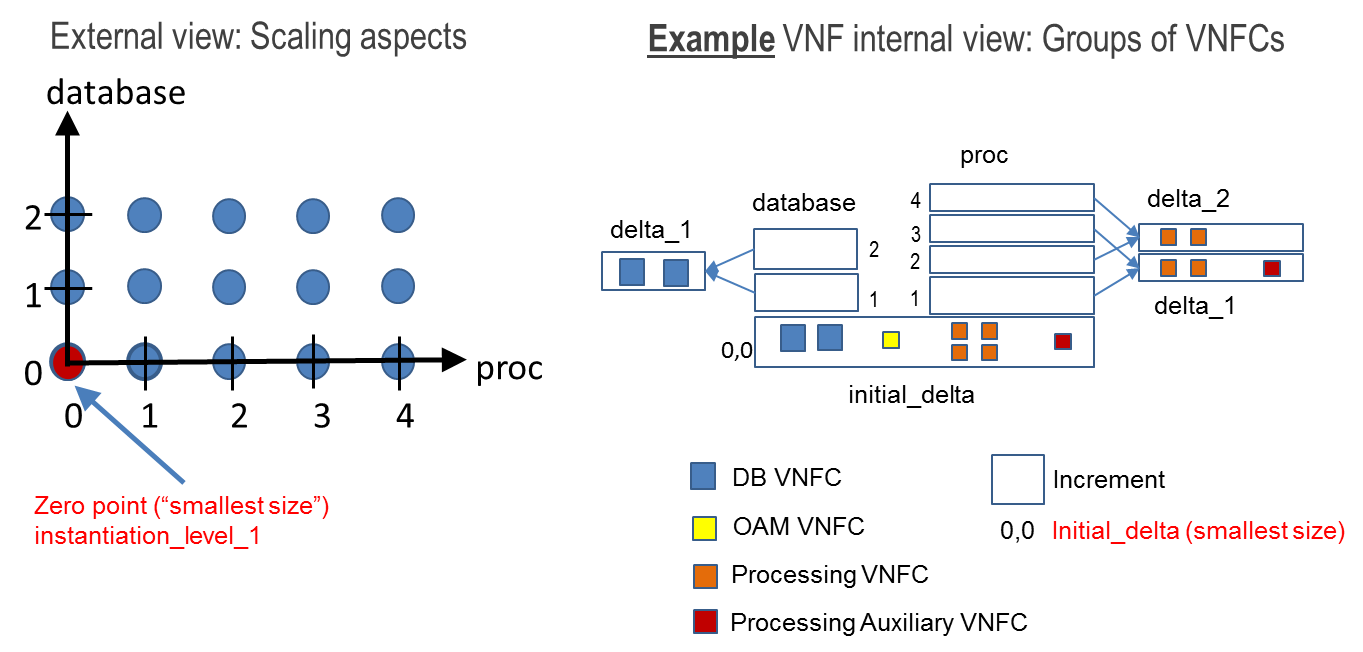


Figure A.6.2-1: Complex scaling example with non-uniform delta

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

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

topology\_template:

node\_templates:

db:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 2

max\_number\_of\_instances: 6

nfvi\_constraints:

..

oam:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 1

nfvi\_constraints:

..

processing:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 4

max\_number\_of\_instances: 12

nfvi\_constraints:

..

processing\_auxiliary:

type: tosca.nodes.nfv.Vdu.Compute

properties:

name: ..

description: ..

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 3

nfvi\_constraints:

..

policies:

- scaling\_aspects:

type: tosca.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: tosca.policies.nfv.InstantiationLevels

properties:

levels:

instantiation\_level\_1:

description: ..

scale\_info:

database:

scale\_level: 0

proc:

scale\_level: 0

- db\_instantiation\_levels:

type: tosca.policies.nfv.VduInstantiationLevels

properties:

levels:

instantiation\_level\_1:

number\_of\_instances: 2

targets: [ db ]

- oam\_instantiation\_levels:

type: tosca.policies.nfv.VduInstantiationLevels

properties:

levels:

instantiation\_level\_1:

number\_of\_instances: 1

targets: [ oam ]

- processing\_instantiation\_levels:

type: tosca.policies.nfv.VduInstantiationLevels

properties:

levels:

instantiation\_level\_1:

number\_of\_instances: 4

targets: [ processing ]

- processing\_auxiliary\_instantiation\_levels:

type: tosca.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.

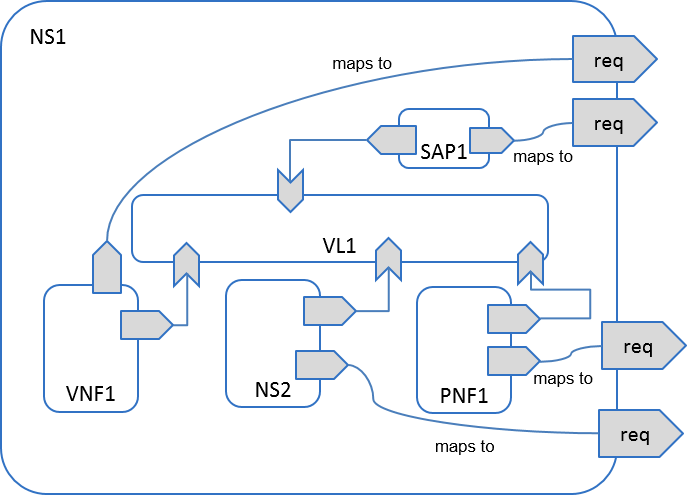


Figure A.7.1-1: Overview of SAP options

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.

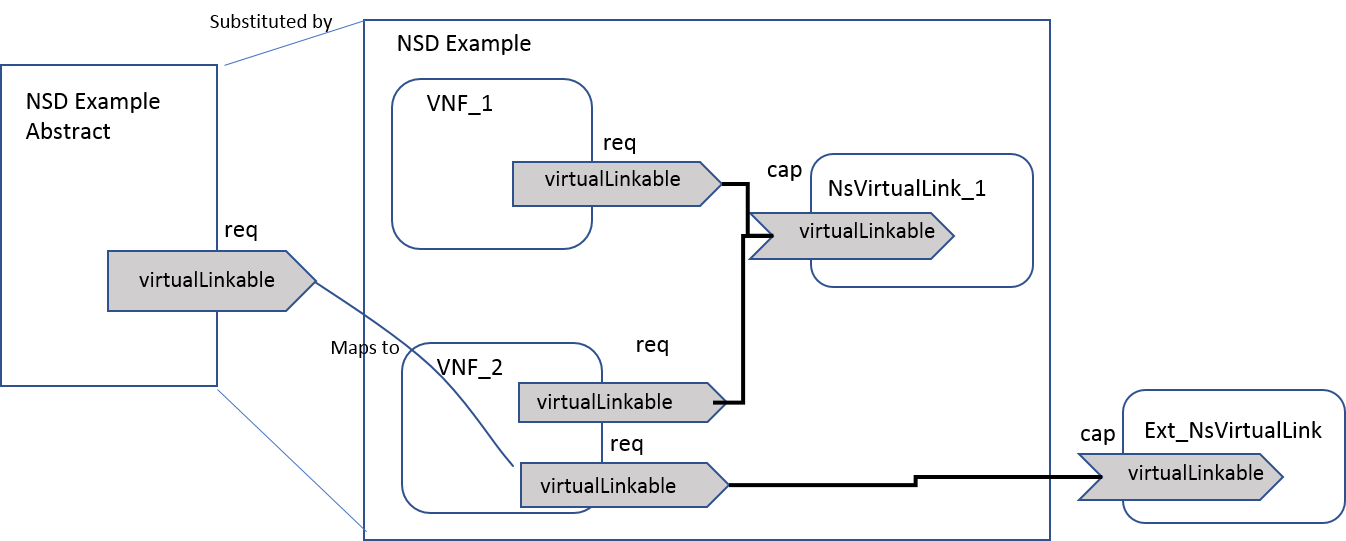


Figure A.7.2-1: VNF\_2 External connection point (VnfExtCp) exposed as a Sap

The following snippet shows the relevant part of the NS service template:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

node\_types:

tosca.nodes.nfv.exampleNS

derived from: tosca.nodes.nfv.NS

# properties:

# properties omitted for brevity

requirements:

- virtual\_link:

capability:

tosca.capabilitites.nfv.VirtualLinkable

...

topology\_template:

...

substitution\_mappings:

node\_type: tosca.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: tosca.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.

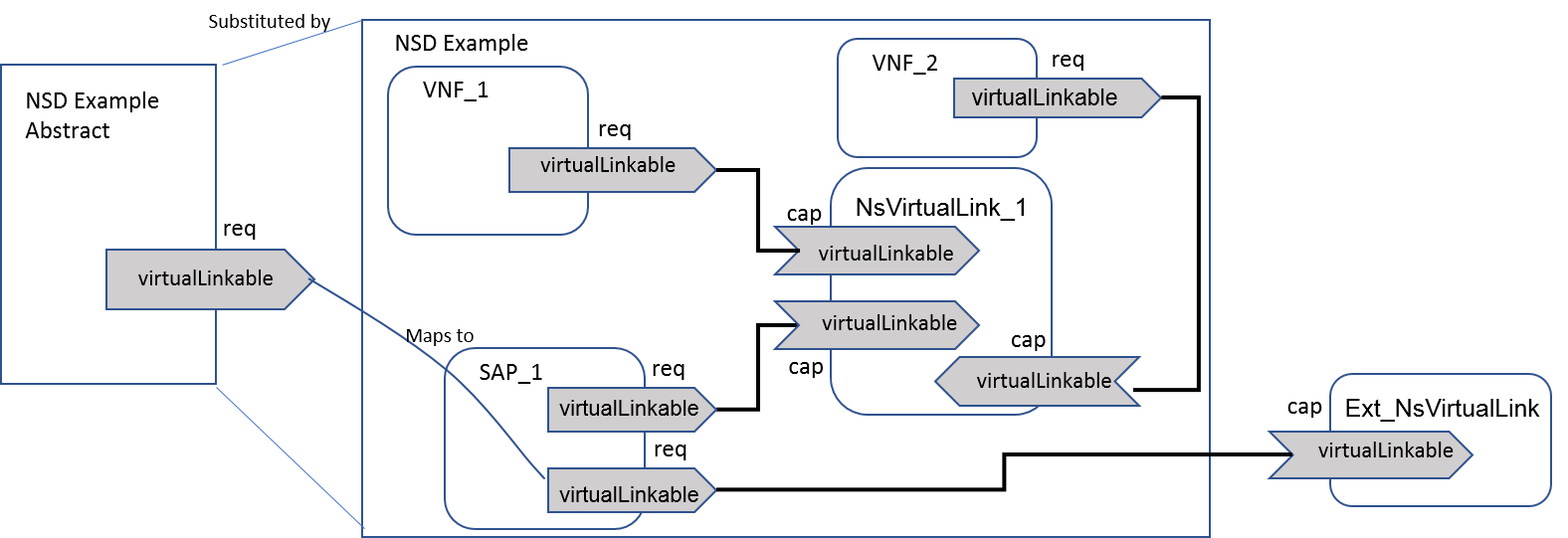


Figure A.7.3-1: SAP\_1 exposed as new CP on an internal virtual link

The following snippet shows the corresponding node template definition:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

node\_types:

tosca.nodes.nfv.exampleNS

derived from: tosca.nodes.nfv.NS

# properties:

# properties omitted for brevity

requirements:

- virtual\_link:

capability:

tosca.capabilitites.nfv.VirtualLinkable

topology\_template:

...

substitution\_mappings:

node\_type: tosca.nodes.nfv.exampleNS

requirements:

virtual\_link: [SAP\_1, external\_virtual\_link] #SAP\_1 is the Sap of the NS

node\_templates:

SAP\_1:

type: tosca.nodes.nfv.Sap

# properties omitted for brevity

...

requirements:

- internal\_virtual\_link: NsVirtualLink\_1

# - external\_virtual\_link: # 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 an NS which 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". In the VNF\_2 node template, it defines a dependency requirement to VNF\_1, which indicates that all the instances of VNF\_1 have to be deployed first before deploy the instances of VNF\_2. NsMonitoring and VnfMonitoring policies represent information to be monitored during the lifetime of a network service and VNF instances.

VNF\_1

NsVirtualLink\_1

Ext\_NsVirtualLink

NS

VNF\_2

Figure A.8-1: example\_NS (simple)

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: Relational database, simple

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of NSD related TOSCA types as defined in ETSI GS NFV-SOL 001

- example\_VNF1.yaml # uri of the yaml file which contains the definition of tosca.nodes.nfv.example\_VNF1, this file might be included in the NSD file structure

- example\_VNF2.yaml # uri of the yaml file which contains the definition of tosca.nodes.nfv.example\_VNF2, this file might be included in the NSD file structure

data\_types:

MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters:

derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters

properties:

parameter\_1:

type: string

required: true

default: value\_1

parameter\_2:

type: string

required: true

default: value\_2

node\_types:

tosca.example\_NS:

derived\_from: tosca.nodes.nfv.NS

properties:

descriptor\_id:

type: string

constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]

default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

designer:

type: string

constraints: [ equal: MyCompany ]

default: MyCompany

name:

type: string

constraints: [ equal: ExampleService ]

default: ExampleService

version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

invariant\_id:

type: string

constraints: [ equal: 1111-2222-aaaa-bbbb ]

default: 1111-2222-aaaa-bbbb

flavour\_id:

type: string

constraints: [ equal: simple ]

default: simple

interfaces:

Nslcm:

type: tosca.interfaces.nfv.Nslcm

instantiate:

inputs:

additional\_parameters:

type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters

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: tosca.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: tosca.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

- dependency: VNF\_1

NsVirtualLink\_1:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

policies:

- my\_service\_NsMonitoring:

type: tosca.policies.nfv.NsMonitoring

properties:

ns\_monitoring\_parameters:

- name: MyService\_byte\_incoming\_sap

performance\_metric: byte\_incoming\_sap

collection\_period: 1 s

- name: MyService\_byte\_outgoing\_sap

performance\_metric: byte\_outgoing\_sap

collection\_period: 1 s

targets: [ my\_service ]

- VNF\_1\_VnfMonitoring:

type: tosca.policies.nfv.VnfMonitoring

properties:

vnf\_monitoring\_parameters:

- name: VNF\_1\_v\_cpu\_usage\_mean\_vnf

performance\_metric: v\_cpu\_usage\_mean\_vnf

collection\_period: 1 s

- name: VNF\_1\_v\_disk\_usage\_mean\_vnf

performance\_metric: v\_disk\_usage\_mean\_vnf

collection\_period: 1 s

targets: [ VNF\_1 ]

- VNF\_2\_VnfMonitoring:

type: tosca.policies.nfv.VnfMonitoring

properties:

vnf\_monitoring\_parameters:

- name: VNF\_2\_v\_cpu\_usage\_mean\_vnf

performance\_metric: v\_cpu\_usage\_mean\_vnf

collection\_period: 1 s

- name: VNF\_2\_v\_disk\_usage\_mean\_vnf

performance\_metric: v\_disk\_usage\_mean\_vnf

collection\_period: 1 s

- name: VNF\_2\_v\_memory\_usage\_mean\_vnf

performance\_metric: v\_memory\_usage\_mean\_vnf

collection\_period: 1 s

targets: [ VNF\_2 ]

The above example illustrates three policies (1 NS-level policy and 2 VNF-level policies). Each of the policies has different monitoring parameters. In the case, where the values of vnf\_monitoring\_parameters are the same across all constituent VNFs, a single VnfMonitoring policy can be used with a targets referencing all VNF node types.

# A.9 Mapping between NFV IM and TOSCA concepts

## A.9.1 Introduction

This clause 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 011 [1] 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 tosca.nodes.nfv.VnfExtCp or tosca.nodes.nfv.VduCp |
| VnfLcmOperationsConfiguration (clause 7.1.5) | property of VNF node type with data type tosca.datatypes.nfv.VnfConfigurableProperties |
| Vdu (clause 7.1.6) | node template with type tosca.nodes.nfv.Vdu.Compute |
| VLd (clause 7.1.7) | node template with type tosca.nodes.nfv.VnfVirtualLink |
| 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 tosca.datatypes.nfv.VduProfile |
| VirtualLinkProfile (clause 7.1.8.4) | property of VnfVirtualLink node type with data type tosca.datatypes.nfv.VlProfile |
| VirtualLinkDescFlavour (clause 7.1.8.5) | Only qos attribute in VirtualLinkDescFlavour has been defined as property of the VlProfile data type |
| InstantiationLevel (clause 7.1.8.7) | policy with type tosca.policies.nfv.InstantiationLevels |
| VduLevel (clause 7.1.8.9) | policy with type tosca.policies.nfv.VduInstantiationLevels |
| LocalAffinityOrAntiAffinityRule (clause 7.1.8.11) | policy with type tosca.policies.nfv.AffinityRule or tosca.policies.nfv.AntiAffinityRule |
| AffinityOrAntiAffinityGroup (clause 7.1.8.12) | policy with type tosca.policies.nfv.AffinityRule or tosca.policies.nfv.AntiAffinityRule |
| 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 tosca.nodes.nfv.Vdu.VirtualBlockStorage |
| VirtualStorageDesc with ObjectStorageData (clause 7.1.9.4.4) | node template with type tosca.nodes.nfv.Vdu.VirtualObjectStorage |
| VirtualStorageDesc with FileStorageData (clause 7.1.9.4.5) | node template with type tosca.nodes.nfv.Vdu.VirtualFileStorage |
| ScalingAspect (clause 7.1.10.2) | policy with type tosca.policies.nfv.ScalingAspects |
| ScalingDelta (clause 7.1.10.4) | policy with type tosca.policies.nfv.VduScalingAspectDeltas and/or tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas |
| VnfIndicator (clause 7.1.11.2) | interface with type tosca.interfaces.nfv.VnfIndicator and related attributes of the VNF node type |
| 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 tosca.datatypes.nfv.VnfConfigurableProperties |
| LifeCycleManagementScript (clause 7.1.13) | interface with type tosca.interfaces.nfv.Vnflcm |
| VnfInfoModifiableAttributes (clause 7.1.14) | property of VNF node template with data type tosca.datatypes.nfv.VnfInfoModifiableAttributes |

## A.9.3 Mapping between ETSI GS NFV-IFA 014 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 ETSI GS NFV-IFA 014 [2] IM and TOSCA concepts

| ETSI GS NFV-IFA 014 [2]  information element | **TOSCA concept** |
| --- | --- |
| NSD (clause 6.2.2) | TOSCA service template(s) in the NSD file structure |
| Sapd (clause 6.2.3) | node template with type tosca.nodes.nfv.Sap or the virtual\_link requirement in the substitution\_mapping |
| MonitoredData (clause 6.2.6) | policy with type tosca.policies.nfv.VnfMonitoring  and tosca.policies.nfv.NsMonitoring |
| VnfIndicatorData (clause 6.2.7) | Not supported in this version of the present document |
| LifeCycleManagementScript (clause 6.2.9) | Interface with type Nslcm |
| NsDf (clause 6.3) | lower level service template(s) in the NSD file structure |
| VnfProfile (clause 6.3.3) | property of VNF node template with data type tosca.datatypes.nfv.VnfProfile |
| VirtualLinkProfile (clause 6.3.4) | property of NsVirtualLink node template with data type tosca.datatypes.nfv.VlProfile |
| AffinityOrAntiAffinityGroup (clause 6.3.5) | policy with type tosca.policies.nfv.NsAffinityRule or tosca.policies.nfv.NsAntiAffinityRule |
| NsVirtualLinkConnectivity (clause 6.3.7) | requirement and capability between PnfExtCp or VnfExtCp and NsVirtualLink 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) | property of NS node template with data type tosca.datatypes.nfv.NsProfile |
| Dependencies (clause 6.3.12) | dependency requirements |
| Vnffgd (clause 6.4.2) | group with type tosca.groups.nfv.VNFFG |
| Nfpd (clause 6.4.3) | node template with type tosca.nodes.nfv.NFP |
| NfpPositionElement (clause 6.4.6) | node template with type tosca.nodes.nfv.NfpPositionElement |
| NfpPositionDesc(clause 6.4.5) | node template with type tosca.nodes.nfv.NfpPosition |
| NsVirtualLinkDesc (clause 6.5.2) | node template with type tosca.nodes.nfv.NsVirtualLink |
| VirtualLinkDf (clause 6.5.4) | Only the serviceAvaibilityLevel and qos attributes in VirtualLinkDf have been defined as properties of the NsVlProfile data type. The flavour id is not mapped to a property as there is only one flavour of a particular NS virtual link per NS service template |
| Pnfd (clause 6.6.2) | A standalone TOSCA service template |
| PnfExtCpd (clause 6.6.4) | node template with type tosca.nodes.nfv.PnfExtCp |
| 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.

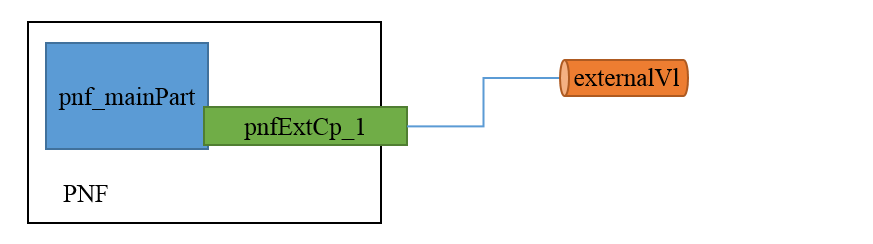


Figure A.10-1: examplePnf

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: the service template of a PNFD

imports:

- etsi\_nfv\_sol001\_pnfd\_types.yaml

node\_types:

MyCompany.examplePnf.1\_0:

derived\_from: tosca.nodes.nfv.PNF

properties:

descriptor\_id:

type: string

constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233 ]

default: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233

function\_description:

type: string

default: an example PNF

provider:

type: string

constraints: [ equal: MyCompany ]

default: MyCompany

version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

descriptor\_invariant\_id:

type: string

constraints: [ equal: 1111-2222-ccaa-bbdd ]

default: 1111-2222-ccaa-bbdd

name:

type: string

constraints: [ equal: ExamplePnf ]

default: ExamplePnf

requirements:

- virtual\_link:

capability: tosca.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: tosca.nodes.nfv.PnfExtCp

properties:

trunk\_mode: false

layer\_protocols: [ ipv4 ]

role: leaf

description: External connection point to access this pnf

requirements:

# - external\_virtual\_link:

- dependency: pnf\_mainPart

# A.11 NSD with Multiple deployment flavour modelling design example

Deployment flavours are represented as deployable TOSCA topology templates. This way one NS service template represents one NS deployment flavour, and different deployment flavours are described by different NS service templates. This is in line with the concept that different deployment flavours can define different topologies of the same NS, with different scaling aspects, different constituent VNFs, PNFs and nested NSs, and different internal connectivity.

In order to represent an NS, a top-level service template is used. This top-level service template contains a topology template with only an abstract NS node which defines the common parts of the different deployment flavours (such as designer, version 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 NS instantiation request which contains a flavour\_id selected among those available in the NSD.

As a result, the NFVO will look into the available further service templates representing the different NS deployment flavours of the NS and use the one that has the matching flavour\_id property value to substitute for the abstract NS. These are the low-level service templates.

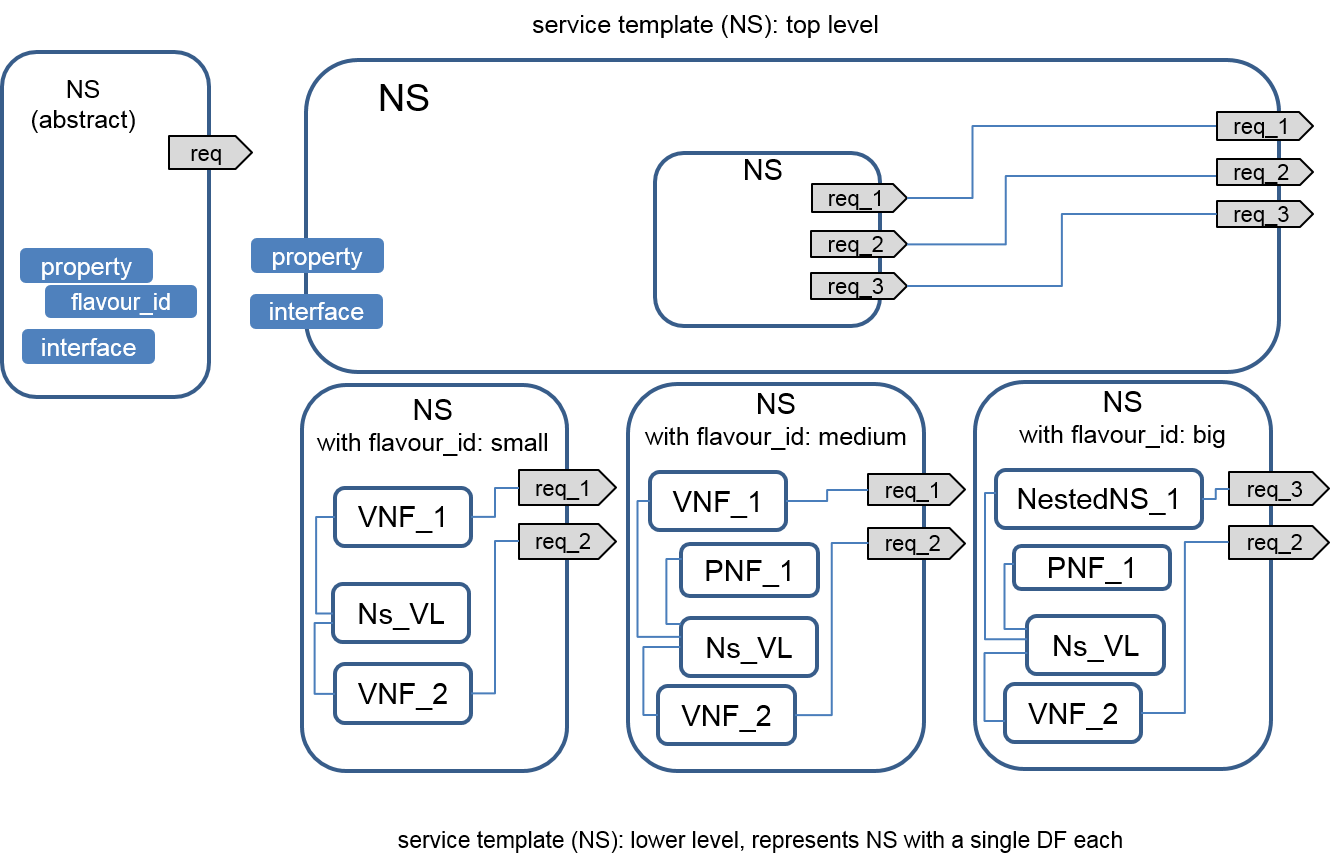


Figure A.11-1: NSD overview with multiple deployment flavour

An NSD contains a TOSCA top-level Service Template as entry point in the NSD file structure and one or more TOSCA low-level Service templates representing the different deployment flavours (see figure A.11-1). The NSD is interpreted by an NFVO. In this example, the templates describe two variants of the NS each corresponding to a deployment flavour: a small and a big one. The small NS consists of two VNFs one NS Virtual link and, the big VNF variant consists of three VNFs and one NS Virtual link.

NSD-top level MyExampleNs.yaml

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: my service

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml

- MyExampleNs\_Type.yaml # contains the NS node type definition

- MyExampleNs\_small.yaml

- MyExampleNs\_big.yaml

topology\_template:

inputs:

flavour\_id:

type: string

description: NS deployment flavour selected by the consumer. It is provided in the SOL005 API

node\_templates:

myexampleNs:

type: tosca.MyExampleNS

properties:

descriptor\_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

designer: MyCompany

name: ExampleService

version: '1.0'

invariant\_id: 1111-2222-aaaa-bbbb

flavour\_id: {get\_input: flavour\_id }

# get\_input function would be used by a TOSCA orchestrator at run time to access the flavour\_id. If the deployment is not done by a TOSCA orchestrator above function may not be needed

# requirements:

#- virtual\_link # mapped in lower-level templates

The MyExampleNs\_Type.yaml file has the following content:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: type definition of tosca.MyExampleNS

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

data\_types:

MyCompany.datatypes.nfv.NsInstantiateNsAdditionalParameters:

derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters

properties:

parameter\_1:

type: string

required: true

default: value\_1

parameter\_2:

type: string

required: true

default: value\_2

node\_types:

tosca.MyExampleNS:

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: [ small, big ] ]

default: small

requirements:

- virtual\_link:

capability: tosca.capabilities.nfv.VirtualLinkable

interfaces:

Nslcm:

type: tosca.interfaces.nfv.Nslcm

instantiate:

inputs:

additional\_parameters:

MyCompany.datatypes.nfv.NsInstantiateNsAdditionalParameters

The NS node template in the **myexample\_NS.yaml** fileis 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 tosca.MyExampleNS with implementation of the LCM interfaces.

**MyExampleNs** (small): Lower level, contains 2 VNFs and 1 NS virtual link.

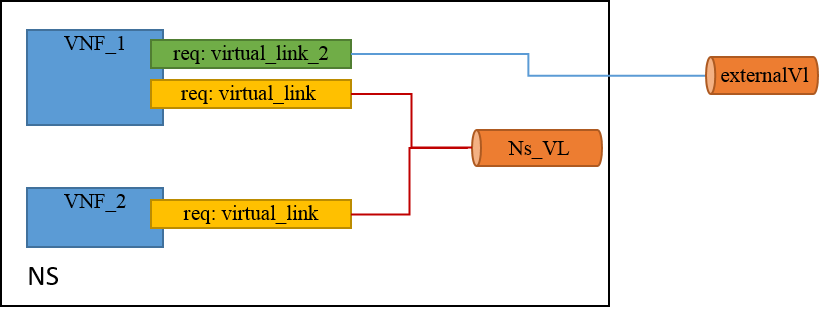


Figure A.11-2: MyExampleNs (simple): Lower level

**MyExampleNs\_small.yaml**

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: myExampleNs with small flavour

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

- MyExampleNs\_Type.yaml # contains the NS node type definition

- example\_VNF1.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF1 node type definition, this file might be included in the NSD file structure

- example\_VNF2.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF2 node type definition, this file might be included in the NSD file structure

topology\_template:

substitution\_mappings:

node\_type: tosca.MyExampleNS

properties:

..

flavour\_id: small

requirements:

virtual\_link: [ VNF\_1, virtual\_link\_2 ]

node\_templates:

MyExampleNS:

type: tosca.MyExampleNS

properties:

#

interfaces:

Nslcm:

instantiate:

implementation: instantiate.workflow.yaml

terminate:

implementation: terminate.workflow.yaml

VNF\_1:

type: tosca.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: Ns\_VL

# - virtual\_link\_2: # map to virtual\_link requirement of the NS node

VNF\_2:

type: tosca.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: Ns\_VL

Ns\_VL:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

**MyExampleNs** (big): Lower level, contains 3 VNFs and 1 NS virtual link.

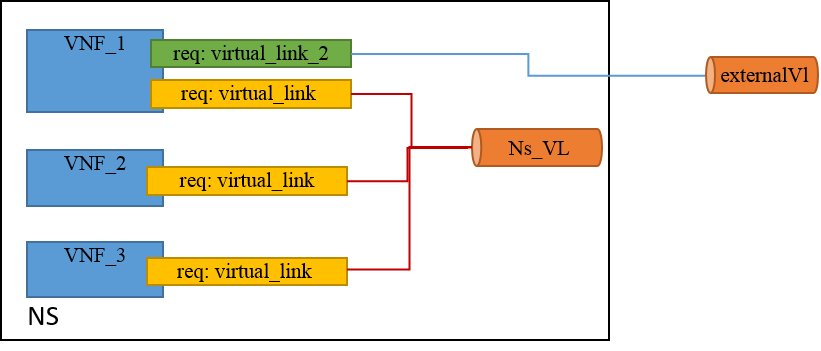


Figure A.11-3: MyExampleNs (big): Lower level

**MyExampleNs\_big.yaml**

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: myExampleNs with big flavour

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

- MyExampleNs\_Type.yaml # contains the NS node type definition

- example\_VNF1.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF1 node type definition, this file might be included in the NSD file structure

- example\_VNF2.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF2 node type definition, this file might be included in the NSD file structure

- example\_VNF3.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF3 node type definition, this file might be included in the NSD file structure

topology\_template:

substitution\_mappings:

node\_type: tosca.MyExampleNS

properties:

..

flavour\_id: big

requirements:

virtual\_link: [ VNF\_1, virtual\_link\_2 ]

node\_templates:

MyExampleNS:

type: tosca.MyExampleNS

properties:

#

interfaces:

Nslcm:

instantiate:

implementation: instantiate.workflow.yaml

terminate:

implementation: terminate.workflow.yaml

scale:

implementation: scale.workbook.yaml

VNF\_1:

type: tosca.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: Ns\_VL

# - virtual\_link\_2: # map to virtual\_link requirement of the NS node

VNF\_2:

type: tosca.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: Ns\_VL

VNF\_3:

type: tosca.nodes.nfv.example\_VNF3

properties:

flavour\_id: simple

vnf\_profile:

instantiation\_level: level\_1

min\_number\_of\_instances: 1

max\_number\_of\_instances: 3

requirements:

- virtual\_link\_1: Ns\_VL

Ns\_VL:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

# A.12 NSD with nested NS design example

A TOSCA service template representing an NSD may contain a node template of some specific NS node type as one of its constituents. The latter is a nested NS. When the containing NS is deployed, the node template of the nested NS is substituted by the topology template representing the nested NS.

Figure A.12-1 illustrates a network service NS\_1 that consists of one VNF (VNF\_1), one NsVirtualLink (NS\_VL\_1\_and one nested NS (NS\_2).

The nested NS consists of two VNFs (VNF\_3 and VNF\_4) and one NsVirtualLink (NS\_VL\_2).

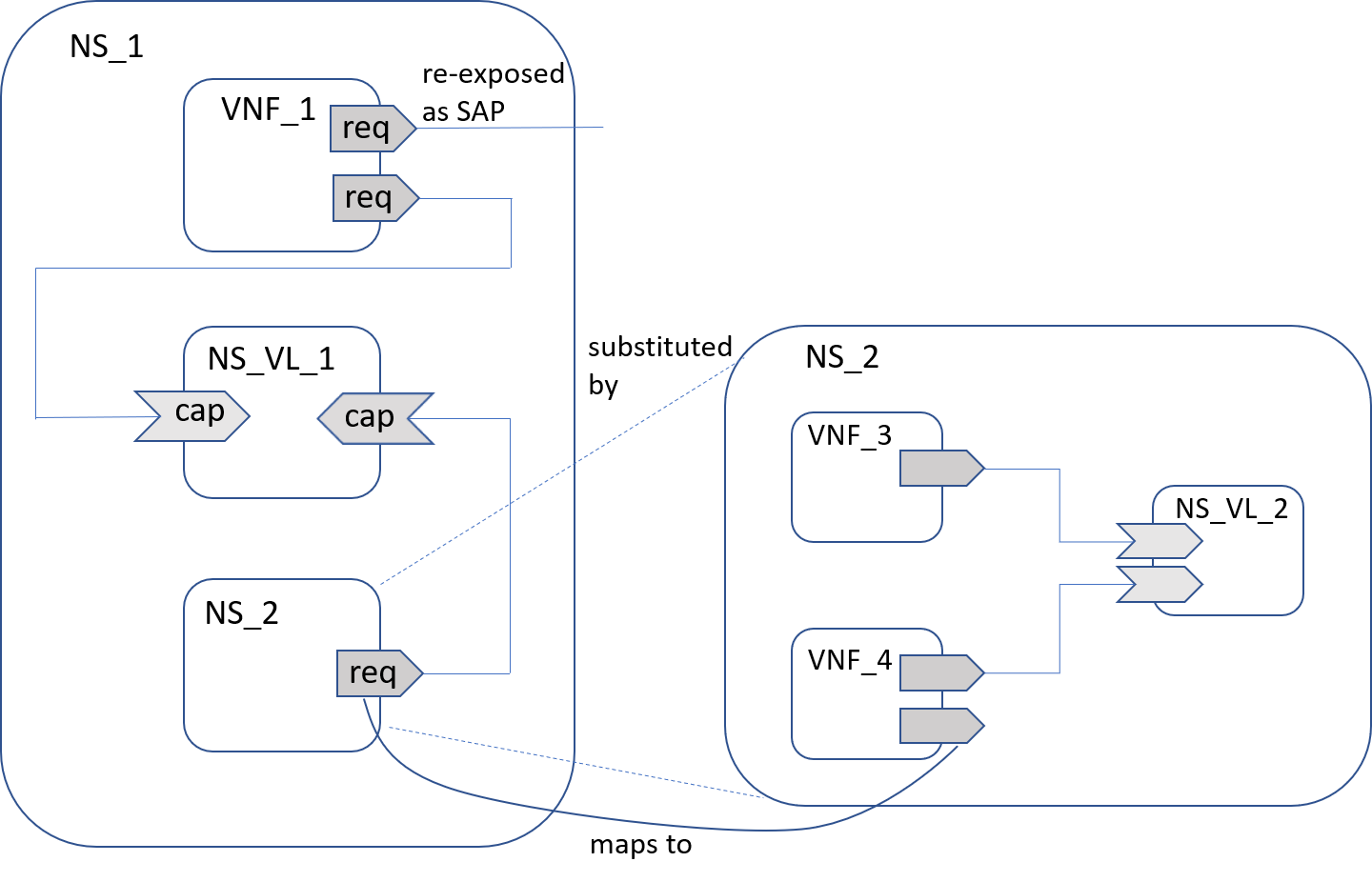


Figure A.12-1: Example of a network service containing a nested network service

The following snipet shows the topology template representing a particular flavour of NS\_1.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: myExampleNs with small flavour

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA NSD types as defined in ETSI GS NFV‑SOL 001

- MyExampleNs\_Type.yaml # contains the NS node type definition

- MyExampleNS\_2.yaml # uri of the yaml file which contains the tosca.myExample.NS\_2 node type definition, this file might be included in the NSD file structure of NS\_1

- example\_VNF1.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF1 node type definition, this file might be included in the NSD file structure of NS\_1

topology\_template:

substitution\_mappings:

node\_type: tosca.MyExampleNS

properties:

..

flavour\_id: small

requirements:

virtual\_link: [ VNF\_1, virtual\_link\_2 ]

node\_templates:

NS\_1:

type: tosca.MyExampleNS

interfaces:

Nslcm:

instantiate:

implementation: instantiate.workflow.yaml

terminate:

implementation: terminate.workflow.yaml

VNF\_1:

type: tosca.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\_1: NS\_VL\_1

# - virtual\_link\_2: # map to virtual\_link requirement of the NS node

NS\_2:

type: tosca.myExample.NS\_2

properties:

descriptor\_id: c1bb0ab8-deab-4fa7-95ed-4840d70a3574

designer: MyCompany

version: 1.0.0.0

name: myExample2Service

invariant\_id: aaaa-bbbb-cccc-dddd

ns\_profile:

ns\_instantiation\_level: level\_1

min\_number\_of\_instances: 1

max\_number\_of\_instances: 3

flavour\_id: simple

NS\_VL\_1:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

The contents of MyExampleNs\_Type.yaml file with the node type definition are as follows:

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: type definition of tosca.MyExampleNS

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of TOSCA types as defined in ETSI GS NFV‑SOL 001

node\_types:

tosca.MyExampleNS:

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: [ small, big ] ]

default: small

requirements:

- virtual\_link:

capability: tosca.capabilities.nfv.VirtualLinkable

interfaces:

Nslcm:

type: tosca.interfaces.nfv.Nslcm

The following snippet shows the service template representing the NSD NS\_2. In this example, NS\_2 supports one single deployment flavour.

**MyExampleNS\_2.yaml:**

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: Relational database, simple

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of NSD related TOSCA types as defined in ETSI GS NFV-SOL 001

- example\_VNF3.yaml # uri of the yaml file which contains the definition of tosca.nodes.nfv.example\_VNF3, this file might be included in the NSD file structure

- example\_VNF4.yaml # uri of the yaml file which contains the definition of tosca.nodes.nfv.example\_VNF4, this file might be included in the NSD file structure

node\_types:

tosca.myExample.NS\_2:

derived\_from: tosca.nodes.nfv.NS

properties:

descriptor\_id:

type: string

constraints: [ valid\_values: [ c1bb0ab8-deab-4fa7-95ed-4840d70a3574 ] ]

default: c1bb0ab8-deab-4fa7-95ed-4840d70a3574

designer:

type: string

constraints: [ valid\_values: [ MyCompany] ]

default: MyCompany

name:

type: string

constraints: [ valid\_values: [ myExample2Service ] ]

default: myExample2Service

version:

type: string

constraints: [ valid\_values: [ '1.0.0.0' ] ]

default: '1.0.0.0'

invariant\_id:

type: string

constraints: [ valid\_values: [ aaaa-bbbb-cccc-dddd ] ]

default: aaaa-bbbb-cccc-dddd

flavour\_id:

type: string

constraints: [ valid\_values: [ simple ] ]

default: simple

topology\_template:

substitution\_mappings:

node\_type: tosca.myExample.NS\_2

requirements:

virtual\_link: [ VNF\_4, virtual\_link\_2 ] # the External connection point of

# VNF\_2 is exposed as the Sap

node\_templates:

NS\_2:

type: tosca.myExample.NS\_2

interfaces:

Nslcm:

instantiate:

implementation: instantiate.workflow.yaml

terminate:

implementation: terminate.workflow.yaml

VNF\_3:

type: tosca.nodes.nfv.example\_VNF3

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: NS\_VL\_2

VNF\_4:

type: tosca.nodes.nfv.example\_VNF4

properties:

flavour\_id: simple

vnf\_profile:

instantiation\_level: level\_1

min\_number\_of\_instances: 1

max\_number\_of\_instances: 3

requirements:

- virtual\_link\_1: NS\_VL\_2

# - virtual\_link\_2: # map to virtual\_link requirement of the NS node

NS\_VL\_2:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

# A.13 Virtual IP address connection point

Virtual IP address connection points (VipCps) are used to allocate one or multiple IP addresses that are shared by other CP instances, which may be instances of the same or of different VduCp or VnfExtCp nodes.

**Load balancing**

In the following example two or more instances of a particular VNFC are created. The respective instances of the VduCp, in addition to their default IP address which is assigned according to the 'protocol' property, share a virtual IP address. The multiple instances are created for load sharing purposes.

In this particular example the VduCp is re-exposed as VnfExtCp. Therefore the VipCp is also re-exposed as VnfExtCp.

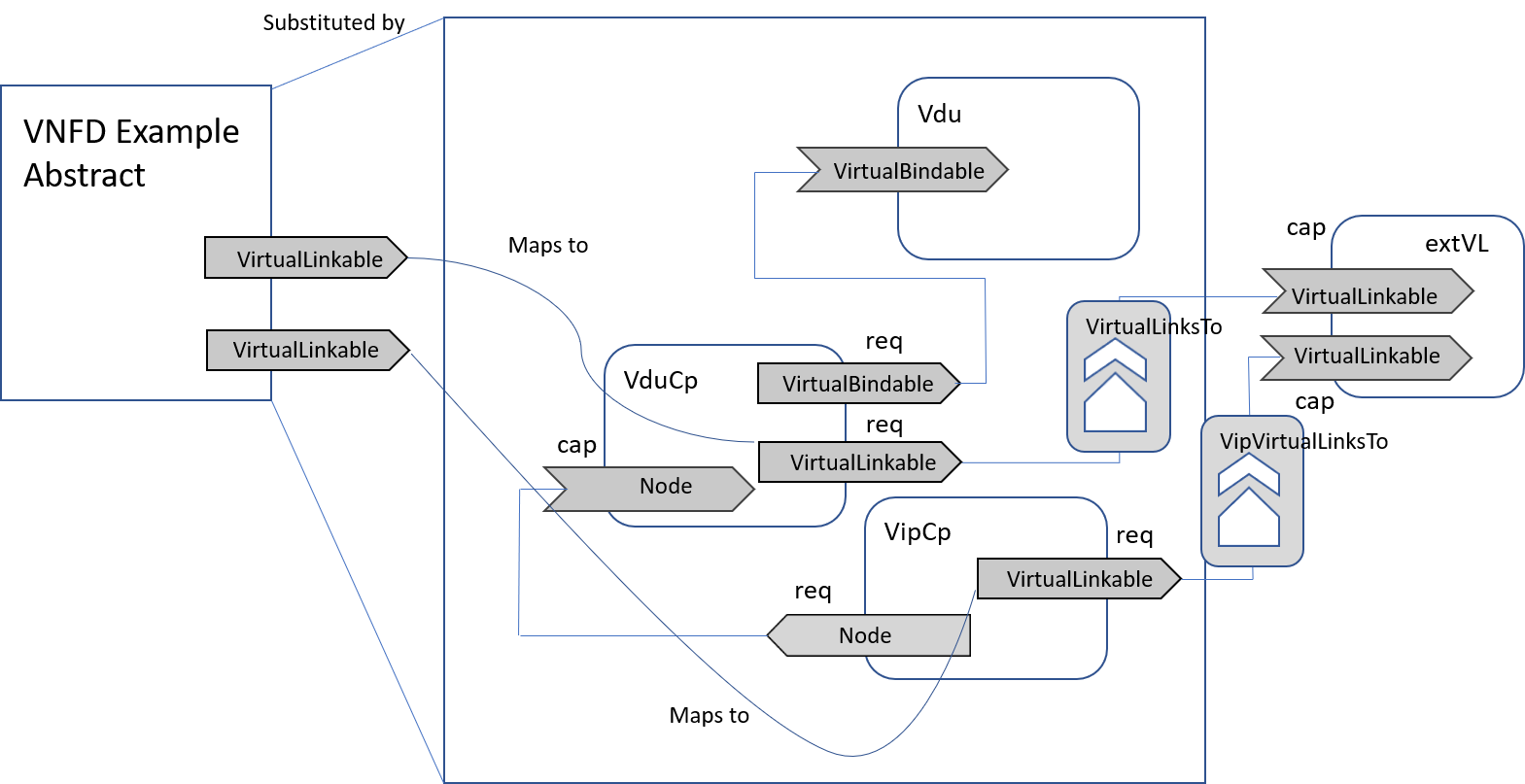


Figure A.13-1: VNFD with a VDU connection point acting as VnfExtCp  
and sharing a virtual IP address

In this example the VduCp and the VipCp are exposed as VnfExtCps. Thus, the VNF abstract node has two requirements for a VirtualLinkable capability. One of them uses the VirtualLinksTo relationship and the other one uses the VipVirtualLinksTo relationship. Both requirements are considered in the substitution mapping.

The following service template shows the relevant parts of the TOSCA VNFD corresponding to figure A.13-1. For simplicity, a single deployment flavour VNF is assumed.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: Relational database, simple

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001 for a VNFD

node\_types:

MyCompany.SunshineDB.1\_0.1\_0:

derived\_from: tosca.nodes.nfv.VNF

properties:

# omitted for brevity

interfaces:

# omitted for brevity

requirements:

- virtual\_link:

capability: tosca.capabilities.nfv.VirtualLinkable

relationship: tosca.relationships.nfv.VirtualLinksTo

occurrences: [ 0, 1 ]  
 - virtual\_link\_vip:

capability: tosca.capabilities.nfv.VirtualLinkable

relationship: tosca.relationships.nfv.VipVirtualLinksTo

occurrences: [ 0, 1 ]

topology\_template:

substitution\_mappings:

node\_type: MyCompany.SunshineDB.1\_0.1\_0

requirements:

virtual\_link: [ Vdu-A-Cp, virtual\_link ]

virtual\_link\_vip: [ VipCp, virtual\_link ]

node\_templates:

SunshineDB:

type: MyCompany.SunshineDB.1\_0.1\_0

properties:

# omitted for brevity

interfaces:

# omitted for brevity

VDU-A:

type: tosca.nodes.nfv.Vdu.Compute

properties:

vdu\_profile:

min\_number\_of\_instances: 2

max\_number\_of\_instances: 5

# other properties omitted for brevity

capabilities:

# omitted for brevity

requirements:

# omitted for brevity

Vdu-A-Cp:

type: tosca.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: VDU-A

#- virtual\_link: # the target node is determined in the NSD

VipCp:

type: tosca.nodes.nfv.VipCp

properties:

vip\_function: load\_balance

protocol:

- associated\_layer\_protocol: ipv4

address\_data:

- address\_type: ip\_address

l3\_address\_data:  
 ip\_address\_assignment: true

floating\_ip\_activated: false

number\_of\_ip\_address: 1

trunk\_mode: false

layer\_protocols: [ ipv4 ]

description: >

Virtual IP connection point. It holds one IP address shared by all instances (between 2 and 5 according to the vdu\_profile) of the Vdu-A-Cp node. Floating IP address is not used in the VipCp. Thus, incoming packets are forwarded with unmodified destination address to one of the instances of Vdu-A-Cp. A router external to the VNF with Equal-Cost Multi-Path (ECMP) load balancing functionality is assumed to be properly configured to route the packets accordingly to the available instances applying load balancing, i.e. one packet is only forwarded to one instance.

requirements:

- target: Vdu-A-Cp

# - virtual\_link: # the target node is determined in the NSD

**High availability**

In the following example, a VNF uses two VNFCs to provide high availability of a service. One of them is the active one receiving IP packets, the other one is in stand-by mode. The VNF logic determines which VNFC is the active and which is the stand-by. The respective VduCp instances, in addition to their default IP address which is assigned according to the 'protocol' property, share a virtual IP address. At any point in time, only one of the VduCp instances, the one belonging to the active VNFC, is bound to the virtual IP address, i.e. only one receives the packets. During the life of the VNF the binding may change, for example in case of failure of the active VNFC, or if determined by the VNF logic. In order to bind the virtual IP address, the active VNFC sends a gratuitous ARP (G-ARP) message with the mapping of the VIP address to its MAC address. A router external to the VNF updates its routing tables when receiving the G-ARP and thereafter routes packets that have the virtual IP address as destination address to the active VNFC.

In this particular example the VduCps are re-exposed as VnfExtCps. Therefore the VipCp is also re-exposed as VnfExtCp.

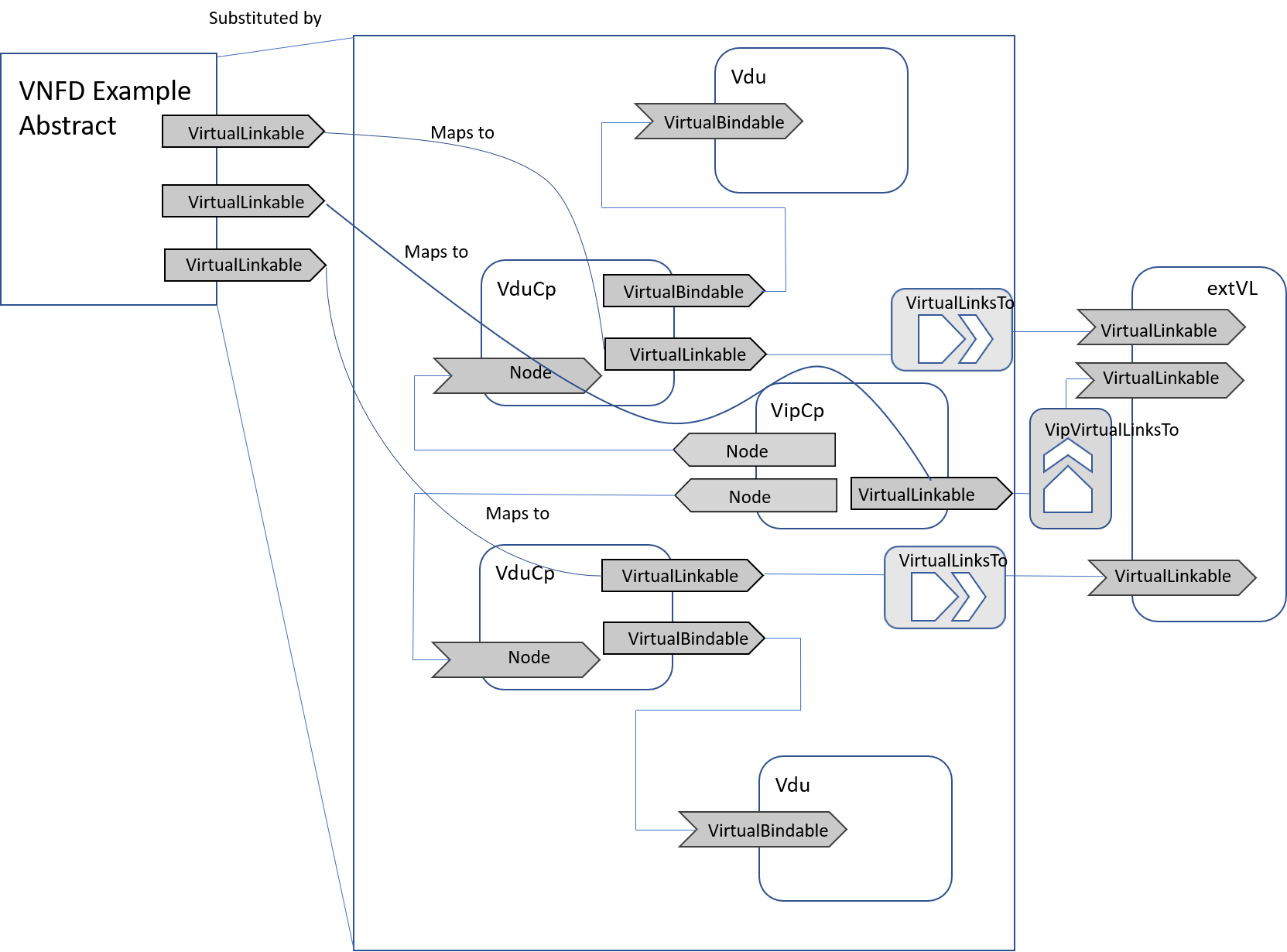


Figure A.13-2: VNFD with two VDU connection points acting as VnfExtCps  
and sharing a virtual IP address

In this example the two VduCps and the VipCp are exposed as VnfExtCps. Therefore the VNF abstract node has three requirements for a VirtualLinkable capability. Two of them use the VirtualLinksTo relationship and the third one uses the VipVirtualLinksTo relationship. The three of them are considered in the substitution mapping.

The following service template shows the relevant parts of the TOSCA VNFD corresponding to figure A.13-2. For simplicity, a single deployment flavour VNF is assumed.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_1

description: Relational database, simple

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001 for a VNFD

node\_types:

MyCompany.SunshineDB.1\_0.1\_0:

derived\_from: tosca.nodes.nfv.VNF

properties:

# omitted for brevity

interfaces:

# omitted for brevity

requirements:

- virtual\_link:

capability: tosca.capabilities.nfv.VirtualLinkable

relationship: tosca.relationships.nfv.VirtualLinksTo

occurrences: [ 0, 1 ]

- virtual\_link\_sby:

capability: tosca.capabilities.nfv.VirtualLinkable

relationship: tosca.relationships.nfv.VirtualLinksTo

occurrences: [ 0, 1 ]  
 - virtual\_link\_vip:

capability: tosca.capabilities.nfv.VirtualLinkable

relationship: tosca.relationships.nfv.VipVirtualLinksTo

occurrences: [ 0, 1 ]

topology\_template:

substitution\_mappings:

node\_type: MyCompany.SunshineDB.1\_0.1\_0

requirements:

virtual\_link: [ Vdu-A-Cp, virtual\_link ]

virtual\_link\_sby: [ Vdu-B-Cp, virtual\_link ]

virtual\_link\_vip: [ VipCp, virtual\_link ]

node\_templates:

SunshineDB:

type: MyCompany.SunshineDB.1\_0.1\_0

properties:

# omitted for brevity

interfaces:

# omitted for brevity

VDU-A:

type: tosca.nodes.nfv.Vdu.Compute

properties:

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 1

# other properties omitted for brevity

capabilities:

# omitted for brevity

requirements:

# omitted for brevity

VDU-B:

type: tosca.nodes.nfv.Vdu.Compute

properties:

vdu\_profile:

min\_number\_of\_instances: 1

max\_number\_of\_instances: 1

# other properties omitted for brevity

capabilities:

# omitted for brevity

requirements:

# omitted for brevity

Vdu-A-Cp:

type: tosca.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: VDU-A

#- virtual\_link: # the target node is determined in the NSD

Vdu-B-Cp:

type: tosca.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: VDU-B

#- virtual\_link: # the target node is determined in the NSD

VipCp:

type: tosca.nodes.nfv.VipCp

properties:

vip\_function: high\_availability

protocol:

- associated\_layer\_protocol: ipv4

address\_data:

- address\_type: ip\_address

l3\_address\_data:  
 ip\_address\_assignment: true

floating\_ip\_activated: true

number\_of\_ip\_address: 1

trunk\_mode: false

layer\_protocols: [ ipv4 ]

description: >

Virtual IP connection point. It holds one IP address shared by the instances of the Vdu-A-Cp and the Vdu-B-Cp nodes (one instance of each). Floating IP address is used. Thus, incoming packets are first NATed to the virtual IP address and then forwarded with the virtual IP address as destination address to the instance of Vdu-A-Cp or Vdu-B-Cp that currently has the address binding.

requirements:

- target: Vdu-A-Cp

- target: Vdu-B-Cp

# - virtual\_link: # the target node is determined in the NSD

In the example above, the VipCp uses a floating IP address. Thus, the incoming packets are expected to have the floating IP as destination address and they are first NATed to the virtual IP address and then forwarded to the instance of the VduCp that currently has the binding to the virtual IP address.

If the VipCp does not use floating IP address, the incoming packets are expected to have the virtual IP address as destination address.

# A.14 NSD with VNF Forwarding Graph design example

The following template fragment illustrates a VNF FG data model for a Network Service. The NS consists of VNF\_1, VNF\_2, VNF\_3 and NsVirtualLink\_1 as its constituents. VNF\_1, VNF\_2 and VNF\_3 node templates have virtual link requirements pointing to node templates of the type tosca.nodes.nfv.Forwarding defined in clause 7.8.8 which in turn have virtual link requirements pointing to the NS virtual links *or to* external virtual links (i.e. *transport links beyond the SAPs*) to which these VNFs are attached.

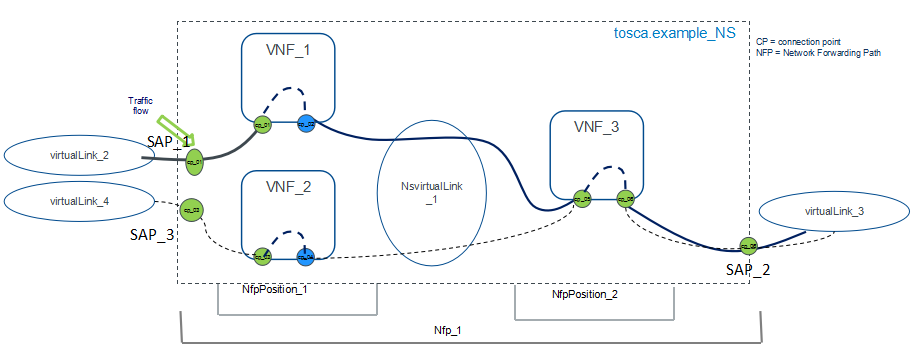


Figure A.14-1: Example Network Forwarding Path

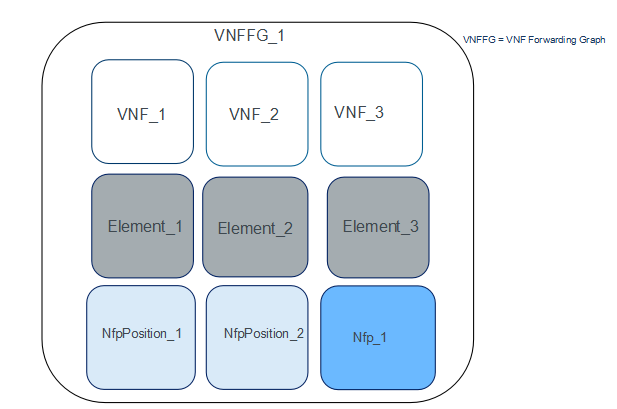


Figure A.14-2: Example VNFFG\_1 group with constituent elements

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2  description: VNF FG Model for example\_NS  imports:  - etsi\_nfv\_sol001\_nsd\_types.yaml # all of NSD related TOSCA types as defined in ETSI GS NFV-SOL 001  - etsi\_nfv\_example\_vnf1.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF1 node type definition, this file might be included in the NSD file structure  - etsi\_nfv\_example\_vnf2.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF2 node type definition, this file might be included in the NSD file structure  - etsi\_nfv\_example\_vnf3.yaml # uri of the yaml file which contains the tosca.nodes.nfv.example\_VNF3 node type definition, this file might be included in the NSD file structure  data\_types:  MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters:  derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters  node\_types:  tosca.example\_NS:  derived\_from: tosca.nodes.nfv.NS  properties:  descriptor\_id:  type: string  constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]  default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177  designer:  type: string  constraints: [ equal: MyCompany ]  default: MyCompany  name:  type: string  constraints: [ equal: ExampleService ]  default: ExampleService  version:  type: string  constraints: [ equal: '1.0' ]  default: '1.0'  invariant\_id:  type: string  constraints: [ equal: 1111-2222-aaaa-bbbb ]  default: 1111-2222-aaaa-bbbb  flavour\_id:  type: string  constraints: [ equal: simple ]  default: simple  interfaces:  Nslcm:  type: tosca.interfaces.nfv.Nslcm  instantiate:  inputs:  additional\_parameters:  type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters  requirements:  - virtual\_link:  capability: tosca.capabilitites.nfv.VirtualLinkable  occurrences: [ 0, 0 ]  - virtual\_link\_2:  capability: tosca.capabilitites.nfv.VirtualLinkable  occurrences: [ 0, 1 ]  - virtual\_link\_3:  capability: tosca.capabilitites.nfv.VirtualLinkable  occurrences: [ 0, 1 ]  - virtual\_link\_4:  capability: tosca.capabilitites.nfv.VirtualLinkable  occurrences: [ 0, 1 ]  topology\_template:  substitution\_mappings:  node\_type: tosca.example\_NS  requirements:  virtual\_link\_2: [ VNF\_1\_forward\_2, virtual\_link ] # the requirement of SAP\_1  virtual\_link\_3: [ VNF\_3\_forward\_6, virtual\_link ] # the requirement of SAP\_2  virtual\_link\_4: [ VNF\_2\_forward\_4, virtual\_link ] # the requirement of SAP\_3  node\_templates:  my\_service:  type: tosca.example\_NS  properties:  descriptor\_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177  designer: MyCompany  name: ExampleService  . . .  interfaces:  Nslcm:  instantiate:  implementation: instantiate.workflow.yaml  terminate:  implementation: terminate.workflow.yaml  NsVirtualLink\_1:  type: tosca.nodes.nfv.NsVirtualLink  properties:  . . .  VNF\_1:  type: tosca.nodes.nfv.example\_VNF1  properties:  descriptor\_id: b1bb0ce7-2222-4fa7-95ed-4840d70a1179  descriptor\_version: "1.0.0"  . . .  requirements:  - virtual\_link: VNF\_1\_forward\_1  - virtual\_link\_2: VNF\_1\_forward\_2  VNF\_1\_forward\_1:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link: NsVirtualLink\_1  VNF\_1\_forward\_2:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link\_2: # map to virtual\_link requirement of the NS node  VNF\_2:  type: tosca.nodes.nfv.example\_VNF2  properties:  descriptor\_id: b1bb0ce7-2222-4fa7-95ed-4840d70a1178  descriptor\_version: "1.0.0"  . . .  requirements:  - virtual\_link: VNF\_2\_forward\_3  - virtual\_link\_4: VNF\_2\_forward\_4  VNF\_2\_forward\_3:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link: NsVirtualLink\_1  VNF\_2\_forward\_4:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link\_2: # map to virtual\_link requirement of the NS node  VNF\_3:  type: tosca.nodes.nfv.example\_VNF3  properties:  descriptor\_id: b1bb0ce7-2222-4fa7-95ed-4840d70a1177  descriptor\_version: "1.0.0"  . . .  requirements:  - virtual\_link: VNF\_3\_forward\_5  - virtual\_link\_3: VNF\_3\_forward\_6  VNF\_3\_forward\_5:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link: NsVirtualLink\_1    VNF\_3\_forward\_6:  type: tosca.nodes.nfv.Forwarding  requirements:  - virtual\_link\_3: # map to virtual\_link requirement of the NS node  # NfpPositionElement (Service Function) for VNF\_1  Element\_1:  type: tosca.nodes.nfv.NfpPositionElement  requirements:  - profile\_element:  node: VNF\_1\_forward\_1  capability: forwarding  - profile\_element:  node: VNF\_1\_ forward\_2  capability: forwarding  # NfpPositionElement (Service Function) for VNF\_2  Element\_2:  type: tosca.nodes.nfv.NfpPositionElement  requirements:  - profile\_element:  node: VNF\_2\_ forward\_3  capability: forwarding  - profile\_element:  node: VNF\_2\_ forward\_4  capability: forwarding  # NfpPositionElement (Service Function) for VNF\_3  Element\_3:  type: tosca.nodes.nfv.NfpPositionElement  requirements:  - profile\_element:  node: VNF\_3\_forward\_5  capability: forwarding  - profile\_element:  node: VNF\_3\_forward\_6  capability: forwarding  # NfpPosition\_1 with Element\_1 and Element\_2 as constituents  NfpPosition\_1:  type: tosca.nodes.nfv.NfpPosition  properties:  forwarding\_behaviour: lb  requirements:  - element: Element\_1  - element: Element\_2  # NfpPosition\_2 with Element\_3 as constituents  NfpPosition\_2:  type: tosca.nodes.nfv.NfpPosition  properties:  forwarding\_behaviour: all  requirements:  - element: Element\_3  Nfp\_1:  type: tosca.nodes.nfv.NFP  requirements:  - nfp\_position: NfpPosition\_1  - nfp\_position: NfpPosition\_2  policies:  - NfpRule\_1:  type: tosca.policies.nfv.NfpRule  properties:  ether\_destination\_address: 00:0a:95:9d:68:16  ether\_source\_address: 00:A0:C9:14:C8:29  ether\_type: ipv4  vlan\_tag:  - 10  - 20  - 30  protocol: tcp  dscp: 101111  source\_port\_range: [ 5000, 15000 ]  destination\_port\_range: [ 800, 8080 ]  source\_ip\_address\_prefix: 10.10.10.0  destination\_ip\_address\_prefix: 125.1.12.111  extended\_criteria:  criteria:  starting\_point: 3  length: 4  value: 1000  targets: [ Nfp\_1 ]  groups:  VNFFG\_1:  type: tosca.groups.nfv.VNFFG  properties:  description: VNF Forwarding Graph for example\_NS  members: [ Nfp\_1, VNF\_1, VNF\_2, VNF\_3, NsVirtualLink\_1, Element\_1, Element\_2, Element\_3 ] |

The following template fragment is part of the content in the etsi\_nfv\_example\_vnf1.yaml.

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2  description: VNF Descriptor for VNF1  imports:  - etsi\_nfv\_sol001\_vnfd\_types.yaml # all of VNFD related TOSCA types as defined in ETSI GS NFV-SOL 001  node\_types:  tosca.nodes.nfv.example\_VNF1:  derived\_from: tosca.nodes.nfv.VNF  properties:  . . . . . .  topology\_template:  substitution\_mappings:  node\_type: tosca.nodes.nfv.example\_VNF1  requirements:  virtual\_link: [ cp\_02, external\_virtual\_link ]  virtual\_link\_2: [ cp\_01, external\_virtual\_link ]  node\_templates:  cp\_01:  type: tosca.nodes.nfv.VnfExtCp  properties:  . . . ......  cp\_02:  type: tosca.nodes.nfv.VnfExtCp  properties:  . . . ...... |

The following template fragment is part of the content in the etsi\_nfv\_example\_vnf2.yaml.

|  |
| --- |
| tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2  description: VNF Descriptor for VNF2  imports:  -.etsi\_nfv\_sol001\_vnfd\_types.yaml # all of VNFD related TOSCA types as defined in ETSI GS NFV-SOL 001  node\_types:  tosca.nodes.nfv.example\_VNF2:  derived\_from: tosca.nodes.nfv.VNF  properties:  . . . . . .  topology\_template:  substitution\_mappings:  node\_type: tosca.nodes.nfv.example\_VNF2  requirements:  virtual\_link: [ cp\_04, external\_virtual\_link ]  virtual\_link\_4: [ cp\_03, external\_virtual\_link ]  node\_templates:  cp\_03:  type: tosca.nodes.nfv.VnfExtCp  properties:  . . . ......  cp\_04:  type: tosca.nodes.nfv.VnfExtCp  properties:  . . . ...... |

The following template fragment is part of the content in the etsi\_nfv\_example\_vnf3.yaml.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_2

description: VNF Descriptor for VNF3

imports:

- etsi\_nfv\_sol001\_vnfd\_types.yaml # all of VNFD related TOSCA types as defined in ETSI GS NFV-SOL 001

node\_types:

tosca.nodes.nfv.example\_VNF3:

derived\_from: tosca.nodes.nfv.VNF

properties:

. . .

topology\_template:

substitution\_mappings:

node\_type: tosca.nodes.nfv.example\_VNF3

requirements:

virtual\_link: [ cp\_05, external\_virtual\_link ]

virtual\_link\_3: [ cp\_06, external\_virtual\_link ]

node\_templates:

cp\_05:

type: tosca.nodes.nfv.VnfExtCp

properties:

. . .

cp\_06:

type: tosca.nodes.nfv.VnfExtCp

properties:

. . .

Annex B (normative):  
etsi\_nfv\_sol001\_type definitions

# B.1 Purpose

All type definitions specified in clauses 6, 7, 8 and 9 of the present document are gathered in four definition files.

The file names are structured as follows:

* etsi\_nfv\_sol001\_common\_types.yaml, for the common type definitions provided in clause 9 which are used by at least two types of deployment templates among those identified in clause 5.1;
* etsi\_nfv\_sol001\_vnfd\_types.yaml, for the definitions provided in clause 6 and only used in a VNFD service template design;
* etsi\_nfv\_sol001\_nsd\_types.yaml, for the definitions provided in clause 7 and only used in an NSD service template design;
* etsi\_nfv\_sol001\_pnfd \_types.yaml, for the definitions provided in clause 8 and only used in a PNFD service template design.

# B.2 VNFD type definitions file

The file etsi\_nfv\_sol001\_vnfd\_types.yaml contains all type definitions specified in clause 6 of the present document and is contained in archive gs\_nfv-sol001v020801p0.zip which accompanies the present document. This file is also available at the following URL:

<https://forge.etsi.org/rep/nfv/SOL001/raw/v2.8.1/etsi_nfv_sol001_vnfd_types.yaml>

NOTE 1: The file etsi\_nfv\_sol001\_vnfd\_types.yaml includes a TOSCA import definition referencing etsi\_nfv\_sol001\_common\_types.yaml file. If the later file is included in the VNF package, the import definition can reference the local file using appropriate path in the VNF package.

This file is a TOSCA service template that only contains definitions. The template\_version in the metadata section within this template is structured as x.y.z, where x, y and z represent the version of this file and are set respectively to "2", "8" and "1" for this version of the present document. In subsequent versions of the present document, "x", "y" and "z" in the template\_version will be incremented only if there are changes in the VNFD type definitions.

A TOSCA service template representing a VNFD complying with the present document shall contain import statement referencing this file, as defined in clause 5.6.1.

NOTE 2: This file may, but need not, be included in the VNF Package.

# B.3 NSD type definitions file

The file etsi\_nfv\_sol001\_nsd\_types.yaml contains all type definitions specified in clause 7 of the present document and is contained in archive gs\_nfv-sol001v020801p0.zip which accompanies the present document. This file is also available at the following URL:

<https://forge.etsi.org/rep/nfv/SOL001/raw/v2.8.1/etsi_nfv_sol001_nsd_types.yaml>

NOTE 1: The file etsi\_nfv\_sol001\_nsd\_types.yaml includes a TOSCA import definition referencing etsi\_nfv\_sol001\_common\_types.yaml file. If the later file is included in the NSD file archive, the import definition can reference the local file using appropriate path in the NSD file archive.

This file is a TOSCA service template that only contains definitions. The template\_version in the metadata section within this template is structured as x.y.z, where x, y and z represent the version of this file and are set respectively to "2", "8" and "1" for this version of the present document. In subsequent versions of the present document, "x", "y" and "z" in the template\_version will be incremented only if there are changes in the NSD type definitions.

A TOSCA service template representing an NSD complying with the present document shall contain import statement referencing this file, as defined in clause 5.6.2.

NOTE 2: 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\_types.yaml contains all type definitions specified in clause 8 of the present document and is contained in archive gs\_nfv-sol001v020801p0.zip which accompanies the present document. This file is also available at the following URL:

<https://forge.etsi.org/rep/nfv/SOL001/raw/v2.8.1/etsi_nfv_sol001_pnfd_types.yaml>

NOTE: The file etsi\_nfv\_sol001\_pnfd\_types.yaml includes a TOSCA import definition referencing etsi\_nfv\_sol001\_common\_types.yaml file.

This file is a TOSCA service template that only contains definitions. The template\_version in the metadata section within this template is structured as x.y.z, where x, y and z represent the version of this file and are set respectively to "2", "8" and "1" for this version of the present document. In subsequent versions of the present document, "x", "y" and "z" in the template\_version will be incremented only if there are changes in the PNFD type definitions.

A TOSCA service template representing a PNFD complying with the present document shall contain import statement referencing this file, as defined in clause 5.6.3.

# B.5 Common type definitions file

The file etsi\_nfv\_sol001\_common\_types.yaml contains the type definitions as specified in clause 9 and used by at least two types of deployment templates and is contained in archive gs\_nfv-sol001v020801p0.zip which accompanies the present document. This file is also available at the following URL:

<https://forge.etsi.org/rep/nfv/SOL001/raw/v2.8.1/etsi_nfv_sol001_common_types.yaml>

This file is a TOSCA service template that only contains definitions. The template\_version in the metadata section within this template is structured as x.y.z, where x, y and z represent the version of this file and are set respectively to "2", "8" and "1" for this version of the present document. In subsequent versions of the present document, "x", "y" and "z" in the template\_version will be incremented only if there are changes in the common type definitions.

NOTE: This file may, but need not, be included in the VNF package or NSD file archive.

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] or [20]. 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] or [20] it uses, unless otherwise stated in clause 6 of the present document.

2) An 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 clauses 6, 7, 8 and 9 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 | PNFD |
| tosca.datatypes.Root | X | X | X |
| tosca.artifacts.Deployment.Image | X |  |  |
| tosca.artifacts.Implementation | X |  |  |
| tosca.capabilities.Root | X | X |  |
| tosca.capabilities.Node | X | X | X |
| tosca.relationships.Root | X | X |  |
| tosca.relationships.DependsOn | X | X | X |
| tosca.interfaces.Root | X | X |  |
| tosca.nodes.Root | X | X | X |
| tosca.groups.Root | X | X |  |
| tosca.policies.Root | X | 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 NSD conformant to the present document shall comply with NSD TOSCA service template design specified in clause 7.11 of the present document.

7) A PNFD conformant to the present document shall comply with PNFD TOSCA service template design specified in clause 8.11 of the present document.

8) A VNFD and NSD conformant to the present document shall comply with rules for Type extension defined in clause 5.7 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, 8 and 9 of the present document.

3) It shall resolve the import definitions as described in clause 5.6 of the present document.

Annex D (informative):  
Mapping between properties of TOSCA types and API attributes

# D.1 Introduction

This annex provides the mapping between properties of TOSCA types defined in the present document and defined in the following API specifications: ETSI GS NFV-SOL 002 [i.4], ETSI GS NFV-SOL 003 [i.9], and ETSI GS NFV‑SOL 005 [i.10].

NOTE: See also annex A "Mapping operations to protocol elements" of ETSI GS NFV-SOL 002 [i.4], ETSI GS NFV-SOL 003 [i.9] and ETSI GS NFV-SOL 005 [i.10] for each operation.

# D.2 VNFD-related constructs

Table D.2-1 provides the mapping between the properties of TOSCA types related to the VNFD and API attributes, which include: resource or notification data type (and referenced structured data type when available), attribute name and type in the resource or notification data type, and the interface operation in which the data type is used.

NOTE: In the "Data model" column of the table, an arrow "->" indicates the navigation through the resource, notification and referenced structured data types.

Table D.2-1: Mapping of API attributes and TOSCA constructs

| ETSI GS NFV-SOL 001 | SOL APIs | | |
| --- | --- | --- | --- |
| Type and Property or entity name | Type and attribute name | Data model | Operation (see note 1) |
| tosca.nodes.nfv.VNF -> descriptor\_id | (Identifier)  vnfdId | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfInfoModificationRequest  VnfInfoModifications | Modify VNF Information  (see SOL002 and SOL003) |
| VnfLcmOpOcc  -> VnfInfoModifications | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> VnfInfoModifications | Notify about VNF LCM  (see SOL002 and SOL003) |
| VnfPkgInfo | Create VNF package  (see SOL005)  Query/Read VNF Package Info  (see SOL003 and SOL005) |
| VnfPackageOnboardingNotification | Notify about VNF Package  (see SOL003 and SOL005) |
| VnfPackageChangeNotification | Notify about VNF Package  (see SOL003 and SOL005) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| UpdateNsRequest  -> InstantiateVnfData | Update NS  (see SOL005) |
| NsLcmOperationOccurrenceNotification  -> AffectedVnf | Notify about NS LCM  (see SOL005) |
| PkgmSubscriptionRequest  -> PkgmNotificationsFilter | Subscription about VNF Package  (see SOL003 and SOL005) |
| PkgmSubscription  -> PkgmNotificationsFilter | Subscription about VNF Package  (see SOL003 and SOL005) |
| tosca.nodes.nfv.VNF -> provider | (String)  vnfProvider | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfPkgInfo | Create VNF package  (see SOL005)  Query/Read VNF Package Info  (see SOL003 and SOL005) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| PkgmSubscriptionRequest  -> PkgmNotificationsFilter  -> vnfProductsFromProviders | Subscription about VNF Package  (see SOL003 and SOL005) |
| PkgmSubscription  -> PkgmNotificationsFilter  -> vnfProductsFromProviders | Subscription about VNF Package  (see SOL003 and SOL005) |
| tosca.nodes.nfv.VNF -> product\_name | (String)  vnfProductName | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfPkgInfo | Create VNF package  (see SOL005)  Query/Read VNF Package Info  (see SOL003 and SOL005) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| PkgmSubscriptionRequest  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts | Subscription about VNF Package  (see SOL003 and SOL005) |
| PkgmSubscription  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts | Subscription about VNF Package  (see SOL003 and SOL005) |
| tosca.nodes.nfv.VNF -> software\_version | (Version) vnfSoftwareVersion | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfPkgInfo | Create VNF package  (see SOL005)  Query/Read VNF Package Info  (see SOL003 and SOL005) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| PkgmSubscriptionRequest  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts  -> versions | Subscription about VNF Package  (see SOL003 and SOL005) |
| PkgmSubscription  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts  -> versions | Subscription about VNF Package  (see SOL003 and SOL005) |
| tosca.nodes.nfv.VNF -> descriptor\_version | (Version)  vnfdVersion | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfInfoModifications | Modify VNF Information  (see SOL002 and SOL003) |
| VnfLcmOpOcc  -> VnfInfoModifications | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> VnfInfoModifications | Notify about VNF LCM  (see SOL002 and SOL003) |
| VnfPkgInfo | Create VNF package  (see SOL005)  Query/Read VNF Package Info  (see SOL003 and SOL005) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| PkgmSubscriptionRequest  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts  -> versions | Subscription about VNF Package  (see SOL003 and SOL005) |
| PkgmSubscription  -> PkgmNotificationsFilter  -> vnfProductsFromProviders  -> vnfProducts  -> versions | Subscription about VNF Package  (see SOL003 and SOL005) |
| tosca.nodes.nfv.VNF -> configurable\_properties | (KeyValuePairs)  vnfConfigurableProperties | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| VnfInfoModificationRequest | Modify VNF Information  (see SOL002 and SOL003) |
| VnfInfoModifications |
| VnfLcmOpOcc  -> VnfInfoModifications | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> VnfInfoModifications | Notify about VNF LCM  (see SOL002 and SOL003) |
| tosca.nodes.nfv.VNF ->flavour\_id | (IdentifierInVnfd)  flavourId | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| InstantiateVnfRequest | Instantiate VNF  (see SOL002 and SOL003) |
| GrantRequest | Grant Lifecycle Operation  (see SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo | Query NS  (see SOL005) |
| (IdentifierInVnfd)  vnfFlavourId | UpdateNsRequest  -> instantiateVnfData | Update NS  (see SOL005) |
| (IdentifierInVnfd)  newFlavourId | ChangeVnfFlavourRequest | Change VNF Flavour  (see SOL002 and SOL003) |
| UpdateNsRequest  -> ChangeVnfFlavourData | Update NS  (see SOL005) |
| tosca.policies.nfv.ScalingAspects  ->aspects[key] | (IdentifierInVnfd) aspectId | VnfInstance  -> instantiatedVnfInfo  -> ScaleInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| ScaleVnfRequest | Scale VNF  (see SOL002 and SOL003) |
| ScaleVnfToLevelRequest  -> ScaleInfo | Scale VNF to level  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> ScaleInfo | Query NS  (see SOL005) |
| tosca.datatypes.nfv.ScaleInfo ->scale\_level | (Integer)  scaleLevel | VnfInstance  -> instantiatedVnfInfo  -> ScaleInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| ScaleVnfToLevelRequest  -> ScaleInfo | Scale VNF to level  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> ScaleInfo | Query NS  (see SOL005) |
| Node template name of type tosca.nodes.nfv.VnfExtCp | (IdentifierInVnfd)  cpdId | VnfInstance  -> instantiatedVnfInfo  -> VnfExtCpInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| InstantiateVnfRequest  -> ExtVirtualLinkData  -> VnfExtCpData | Instantiate VNF  (see SOL002 and SOL003) |
| ChangeVnfFlavourRequest  -> ExtVirtualLinkData  -> VnfExtCpData | Change VNF Flavour  (see SOL002 and SOL003) |
| ChangeExtVnfConnectivityRequest  -> ExtVirtualLinkData  -> VnfExtCpData | Change External VNF Connectivity  (see SOL002 and SOL003) |
| Grant  ->ExtVirtualLinkData  ->VnfExtCpData | Grant Lifecycle Operation  (see SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> VnfExtCpInfo | Query NS  (see SOL005) |
| (IdentifierInVnfd)  resourceTemplateId | GrantRequest  -> ResourceDefinition | Grant Lifecycle Operation  (see SOL003) |
| Node template name of type tosca.nodes.nfv.VnfVirtualLink | (IdentifierInVnfd)  vnfVirtualLinkDescId | VnfInstance  -> instantiatedVnfInfo  -> ExtManagedVirtualLinkInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfInstance  -> instantiatedVnfInfo  -> VnfVirtualLinkResourceInfo | Query VNF  (see SOL003) |
| InstantiateVnfRequest  -> ExtManagedVirtualLinkData | Instantiate VNF  (see SOL002 and SOL003) |
| ChangeVnfFlavourRequest  -> ExtManagedVirtualLinkData | Change VNF Flavour  (see SOL002 and SOL003) |
| VnfLcmOpOcc  -> resourceChanges  -> AffectedVirtualLink | Get Operation Status  (see SOL002 and SOL003) |
| Grant  -> ExtManagedVirtualLinkData | Grant Lifecycle Operation  (see SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> ExtManagedVirtualLinkInfo | Query NS  (see SOL005) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> VnfVirtualLinkResourceInfo | Query NS  (see SOL005) |
| (IdentifierInVnfd)  vnfVirtualLinkDescId | UpdateNsRequest  -> InstantiateVnfData  -> ExtManagedVirtualLinkData | Update NS  (see SOL005) |
| tosca.datatypes.nfv.VnfMonitoringParameter | (IdentifierInVnfd)  id | VnfInstance  -> instantiatedVnfInfo  -> MonitoringParameter | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> MonitoringParameter | Query NS  (see SOL005) |
| tosca.datatypes.nfv.VnfMonitoringParameter ->name | (String)  name | VnfInstance  -> instantiatedVnfInfo  -> MonitoringParameter | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> MonitoringParameter | Query NS  (see SOL005) |
| tosca.nodes.nfv.VNF ->localization\_languages | (String)  localizationLanguage | VnfInstance  -> instantiatedVnfInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo | Query NS  (see SOL005) |
| Node template name of type tosca.nodes.nfv.Vdu.Compute | (IdentifierInVnfd)vduId | VnfInstance  -> instantiatedVnfInfo  -> VnfcResourceInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfInstance  -> instantiatedVnfInfo  -> VnfcInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfLcmOpOcc  -> resourceChanges  -> AffectedVnfc | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> AffectedVnfc | Notify about VNF LCM  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> VnfcResourceInfo | Query NS  (see SOL005) |
| (IdentifierInVnfd)  vnfdVirtualComputeDescId | Grant  -> vimAssets  -> VimConputeResourceFlavour | Grant Lifecycle Operation  (see SOL003) |
| Node template name of type tosca.nodes.nfv.VduCp | (IdentifierInVnfd)  cpdId | VnfInstance  -> instantiatedVnfInfo  -> VnfcResourceInfo  -> vnfcCpInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfInstance  -> instantiatedVnfInfo  -> VnfExtCpInfo  See note 3. | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| InstantiateVnfRequest  -> ExtVirtualLinkData  -> VnfExtCpData  See note 3. | Instantiate VNF  (see SOL002 and SOL003) |
| ChangeVnfFlavourRequest  -> ExtVirtualLinkData  -> VnfExtCpData  See note 3. | Change VNF Flavour  (see SOL002 and SOL003) |
| ChangeExtVnfConnectivityRequest  -> ExtVirtualLinkData  -> VnfExtCpData  See note 3. | Change External VNF Connectivity  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> VnfExtCpInfo  See note 3. | Query NS  (see SOL005) |
| (IdentifierInVnfd)  resourceTemplateId  See note 3. | GrantRequest  -> ResourceDefinition  See note 3. | Grant Lifecycle Operation  (see SOL003) |
| Node template name of type tosca.nodes.nfv.Vdu.VirtualBlockStorage   Node template name of type tosca.nodes.nfv.Vdu.VirtualObjectStorage  Node template name of type tosca.nodes.nfv.Vdu.VirtualFileStorage | (IdentifierInVnfd)  virtualStorageDescId | VnfInstance  -> instantiatedVnfInfo  -> VirtualStorageResourceInfo | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| VnfLcmOpOcc  -> resourceChanges  -> AffectedVirtualStorage | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> AffectedVirtualStorage | Notify about VNF LCM  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance  -> instantiatedVnfInfo  -> VirtualStorageResourceInfo | Query NS  (see SOL005) |
| (IdentifierInVnfd)  resourceTemplateId | GrantRequest  -> ResourceDefinition | Grant Lifecycle Operation  (see SOL003) |
| tosca.datatypes.nfv.VnfInfoModifiableAttributes ->metadata | (KeyValuePairs)  metadata | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| CreateVnfRequest | Create VNF Identifier  (see SOL002 and SOL003) |
| VnfInfoModificationRequest | Modify VNF Information  (see SOL002 and SOL003) |
| VnfInfoModifications |
| VnfLcmOpOcc  -> VnfInfoModifications | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> VnfInfoModifications | Notify about VNF LCM  (see SOL002 and SOL003) |
| tosca.datatypes.nfv.VnfInfoModifiableAttributes ->extensions | (KeyValuePairs)  extensions | VnfInstance | Create VNF Identifier  (see SOL002 and SOL003)  Query VNF  (see SOL002 and SOL003) |
| NsInstance  -> VnfInstance | Query NS  (see SOL005) |
| InstantiateVnfRequest | Instantiate VNF  (see SOL002 and SOL003) |
| VnfInfoModificationRequest | Modify VNF Information  (see SOL002 and SOL003) |
| VnfInfoModifications |
| VnfLcmOpOcc  -> VnfInfoModifications | Get Operation Status  (see SOL002 and SOL003) |
| VnfLcmOperationOccurrenceNotification  -> VnfInfoModifications | Notify about VNF LCM  (see SOL002 and SOL003) |
| tosca.policies.nfv.InstantiationLevel ->levels[key] | (IdentifierInVnfd)  instantiationLevelId | InstantiateVnfRequest | Instantiate VNF  (see SOL002 and SOL003) |
| ScaleVnfToLevelRequest | Scale VNF to level  (see SOL002 and SOL003) |
| ChangeVnfFlavourRequest | Change VNF Flavour  (see SOL002 and SOL003) |
| GrantRequest | Grant Lifecycle Operation  (see SOL003) |
| UpdateNsRequest  -> changeVnfFlavourData  See note 4 | Update NS  (see SOL005) |
| (IdentifierInVnfd)  vnfInstantiationLevelId | UpdateNsRequest  -> InstantiateVnfData  See note 4 | Update NS  (see SOL005) |
| ScaleNsRequest  -> ScaleVnfData  -> ScaleToLevelData | Scale NS  (see SOL005) |
| Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the instantiate operation of the Vnflcm interface.  See note 2. | (KeyValuePairs)  additionalParams | InstantiateVnfRequest | Instantiate VNF  (see SOL002 and SOL003) |
| Properties of  tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the scale operation of the Vnflcm interface  See note 2. | (KeyValuePairs)  additionalParams | ScaleVnfRequest | Scale VNF  (see SOL002 and SOL003) |
| Properties of  tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the 'scale to level' operation of the Vnflcm interface  See note 2. | (KeyValuePairs)  additionalParams | ScaleVnfToLevelRequest | Scale VNF to level  (see [i.4] and [i.9]) |
| Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the 'change vnf flavour operation' of the Vnflcm interface  See note 2. | (KeyValuePairs)  additionalParams | ChangeVnfFlavourRequest | Change VNF Flavour  (see SOL002 and SOL003) |
| Properties of  tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the operate operation of the Vnflcm interface  See note 2. | (KeyValuePairs)  additionalParams | OperateVnfRequest | Operate VNF  (see SOL002 and SOL003) |
| Properties of  tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the heal operation of the Vnflcm interface  See note 2. | (KeyValuePairs)  additionalParams | HealVnfRequest | Heal VNF  (see SOL002 and SOL003) |
| Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the 'change external vnf connectivity' operation of the Vnflcm interface.  See note 2. | (KeyValuePairs)  additionalParams | ChangeExtVnfConnectivityRequest | Change External VNF Connectivity  (see SOL002 and SOL003) |
| Properties of  tosca.datatypes.nfv.VnfOperationAdditionalParameters  in the inputs of the terminate operation of the Vnflcm interface.  See note 2 | (KeyValuePairs)  additionalParams | TerminateVnfRequest | Terminate VNF  (see SOL002 and SOL003) |
| name of type tosca.nodes.nfv.Vdu.Compute  Node template name of type tosca.nodes.nfv.Vdu.VirtualBlockStorage | (IdentifierInVnfd)  id | VnfPkgInfo  -> VnfPackageSoftwareImageInfo | Create VNF Package Info  (see SOL005)  Query VNF Package Info  (see SOL005) |
| Name of attribute for VNF indicator in VNF node | (IdentifierInVnfd)  Id | VnfIndicator | Get indicator value  (see SOL002 and SOL003) |
| VnfIndicatorValueChangeNotification | Notify about VNF indicator value change (see SOL002 and SOL003). |
| Name of attribute for VNF indicator in VNF | (String)  Name | VnfIndicator | Get indicator value  (see SOL002 and SOL003) |
| VnfIndicatorValueChangeNotification | Notify about VNF indicator value change (see SOL002 and SOL003). |
| Value of attribute for VNF indicator in VNF node | (Object)  value | VnfIndicator | Get indicator value  (see SOL002 and SOL003) |
| VnfIndicatorValueChangeNotification | Notify about VNF indicator value change (see SOL002 and SOL003). |
| Node template name of type tosca.nodes.nfv.Vdu.Compute  Node template name of type tosca.nodes.nfv.Vdu.VirtualBlockStorage | (IdentifierInVnfd)  vnfdSoftwareImageId | Grant  -> vimAssets  -> VimSoftwareImage | Grant Lifecycle Operation  (see SOL003) |
| NOTE 1: The entry "SOL002" in the "Operation" column refers to ETSI GS NFV-SOL 002 [i.4], "SOL003" refers to ETSI GS NFV-SOL 003 [i.9] and "SOL005" refers to ETSI GS NFV-SOL 005 [i.10].  NOTE 2: This is an empty base type to be extended to a VNF specific type per LCM operation. The extended VNF specific and LCM operation specific type is the one that actually maps to the additionalParams in the API.  NOTE 3: Only when VduCp is re-exposed as VnfExtCp.  NOTE 4: Additional mappings corresponding to input parameter sets of VNF LCM operations in UpdateNS requests may be added in future versions of the present document. | | | |

# D.3 NSD-related constructs

Table D.3-1 provides the mapping between the properties of TOSCA types related to the NSD and API attributes, which include: resource or notification data type (and referenced structured data type when available), attribute name and type in the resource or notification data type, and the interface operation in which the data type is used.

NOTE: In the "Data model" column of the table, an arrow "->" indicates the navigation through the resource, notification and referenced structured data types.

Table D.3-1: Mapping of API attributes and TOSCA constructs

| SOL001 | SOL APIs | | | |
| --- | --- | --- | --- | --- |
| **Type and Property or entity name** | **Type and attribute name** | **Data model** | **Operation (see note 1)** |
| tosca.nodes.nfv.NS -> descriptor\_id | (Identifier)  nsdIds | LccnSubscriptionRequest -> LifecycleChangeNotificationsFilter -> NsInstanceSubscriptionFilter | Subscription to NS LCM notifications  (see SOL005) |
| (Identifier)  nsdId | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdOnboardingNotification | Notification about NSD management  (see SOL005) |
| NsdOnboardingFailureNotification |
| NsdChangeNotification |
| NsdDeletionNotification |
| NsdmSubscriptionRequest -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription -> NsdmNotificationsFilter |
| CreateNsRequest | Create NS  (see SOL005) |
| NsInstance | Create NS  Query NS  Delete NS  (see SOL005) |
| NsLcmOpOcc -> AffectedNs | Query/read about NS LCM operation occurrence  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedNs | Notification about NS LCM  (see SOL005) |
| (Identifier)  newNsdId | UpdateNsRequest -> AssocNewNsdVersionData | Update NS  (see SOL005) |
| tosca.nodes.nfv.NS -> name | (String)  nsdName | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdmSubscriptionRequest -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription -> NsdmNotificationsFilter |
| tosca.nodes.nfv.NS -> version | (Version)  nsdVersion | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdmSubscriptionRequest -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription -> NsdmNotificationsFilter |
| tosca.nodes.nfv.NS -> designer | (String)  nsdDesigner | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdmSubscriptionRequest -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription -> NsdmNotificationsFilter |
| tosca.nodes.nfv.NS -> invariant\_id | (Identifier)  nsdInvariantId | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdmSubscriptionRequest -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription -> NsdmNotificationsFilter |
| tosca.nodes.nfv.NS -> flavour\_id | (IdentifierInNsd)  flavourId | NsInstance | Create NS  Query NS  Delete NS  (see SOL005) |
| (IdentifierInNsd)  nsFlavourId | InstantiateNsRequest | Instantiate NS  (see SOL005) |
| (IdentifierInNsd)  newNsFlavourId | UpdateNsRequest -> ChangeNsFlavourData | Update NS  (see SOL005) |
| Node template name of type tosca.nodes.nfv.NsVirtualLink | (IdentifierInNsd)  nsVirtualLinkDescId | NsLcmOperationOccurrenceNotification -> AffectedVirtualLink | Notification about NS LCM  (see SOL005) |
| NsInstance -> NsVirtualLinkInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| (No related property in type "tosca.datatype.nfv.NsVlProfile")  See note 2 | (IdentifierInNsd)  nsVirtualLinkProfileId | NsInstance -> NsVirtualLinkInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| (IdentifierInNsd)  vlProfileId | NsLcmOperationOccurrenceNotification -> AffectedVirtualLink | Notification about NS LCM  (see SOL005) |
| See note 2 | (IdentifierInNsd)  nsInstantiationLevelId | InstantiateNsRequest | Instantiate NS  (see SOL005) |
| (IdentifierInNsd)  instantiationLevelId | UpdateNsRequest -> ChangeNsFlavourData | Update NS  (see SOL005) |
| (IdentifierInNsd)  nsInstantiationLevel | ScaleNsRequest -> ScaleNsData -> ScaleNsToLevelData | Scale NS  (see SOL005) |
| Node template name in NSD of type derived from tosca.nodes.nfv.NS | (IdentifierInNsd)  nsProfileId | UpdateNsRequest -> NestedNsInstanceData | Update NS  (see SOL005) |
| InstantiateNsRequest -> NestedNsInstanceData | Instantiate NS  (see SOL005) |
| InstantiateNsRequest -> ParamsForNestedNs |
| Node template name in NSD of type derived from tosca.nodes.nfv.VNF | (IdentifierInNsd)  vnfProfileId | NsLcmOpOcc -> AffectedVnf | Query/read information about NS LCM  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedVnf | Notification about NS LCM  (see SOL005) |
| NsInstance -> AffinityOrAntiAffinityRule | Create NS  Query NS  Delete NS  (see SOL005) |
| InstantiateNsRequest -> AffinityOrAntiAffinityRule | Instantiate NS  (see SOL005) |
| InstantiateNsRequest -> ParamsForVnf |
| InstantiateNsRequest -> VnfLocationConstraint |
| InstantiateNsRequest -> VnfInstanceData |
| ScaleNsRequest -> ScaleNsData -> VnfLocationConstraint | Scale NS  (see SOL005) |
| ScaleNsRequest -> ScaleNsData -> VnfInstanceData |
| UpdateNsRequest -> VnfInstanceData | Update NS  (see SOL005) |
| tosca.nodes.nfv.PNF -> descriptor\_id | (Identifier)  pnfdIds | LccnSubscriptionRequest -> LifecycleChangeNotificationsFilter -> NsInstanceSubscriptionFilter | Subscription to NS LCM  (see SOL005) |
| (Identifier)  pnfdId | PnfdOnboardingNotification | Notification about NSD management  (see SOL005) |
| PnfdOnboardingFailureNotification |
| PnfdDeletionNotification |
| NsLcmOpOcc -> AffectedPnf | Query/read information about NS LCM  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedPnf | Notification about NS LCM  (see SOL005) |
| NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| InstantiateNsRequest -> AddPnfData | Instantiate NS  (see SOL005) |
| UpdateNsRequest -> AddPnfData | Update NS  (see SOL005) |
| tosca.nodes.nfv.PNF -> name | (String)  pnfdName | NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| tosca.nodes.nfv.PNF -> version | (Version)  pnfdVersion | NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| tosca.nodes.nfv.PNF -> provider | (String)  pnfdProvider | NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| tosca.nodes.nfv.PNF -> descriptor\_invariant\_id | (Identifier)  pnfdInvariantId | NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| Node template name in NSD of type derived from tosca.nodes.nfv.PNF | (IdentifierInNsd)  pnfProfileId | NsLcmOpOcc -> AffectedPnf | Query/read information about NS LCM  Fail NS LCM  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedPnf | Notification about NS LCM  (see SOL005) |
| NsInstance -> PnfInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| InstantiateNsRequest -> AddPnfData | Instantiate NS  (see SOL005) |
| UpdateNsRequest -> AddPnfData | Update NS  (see SOL005) |
| Node template name of type tosca.nodes.nfv.Cp | (IdentifierInNsd)  cpdId | InstantiateNsRequest -> AddPnfData -> PnfExtCpData | Instantiate NS  (see SOL005) |
| UpdateNsRequest -> AddPnfData -> PnfExtCpData | Update NS  (see SOL005) |
| UpdateNsRequest -> ModifyPnfData -> PnfExtCpData |
| NsInstance -> PnfInfo -> PnfExtCpInfo | Create NS  Query NS  Delete NS  (see SOL005) |
| Group template name of type tosca.groups.nfv.VNFFG | (IdentifierInNsd)  vnffgdId | NsLcmOpOcc -> AffectedVnffg | Query/read information about NS LCM  Fail NS LCM  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedVnffg | Notification about NS LCM  (see SOL005) |
| UpdateNsRequest -> AddVnffgData | Update NS  (see SOL005) |
| UpdateNsRequest -> UpdateVnffgData -> VnffgInfo |
| See note 2 | (IdentifierInNsd)  aspectId | ScaleNsRequest -> ScaleNsData -> ScaleNsByStepsData | Scale NS  (see SOL005) |
| (IdentifierInNsd)  nsScalingAspectId | ScaleNsRequest -> ScaleNsData -> ScaleNsToLevelData -> NsScaleInfo |
| Name of the policy of type tosca.policies.nfv.NsMonitoring | (IdentifierInNsd)  id | NsInstance -> NsMonitoringParameter | Create NS  Query NS  Delete NS  (see SOL005) |
| tosca.policies.nfv.NsMonitoring  -> ns\_monitoring\_parameters  -> name | (String)  name | NsInstance -> NsMonitoringParameter | Create NS  Query NS  Delete NS  (see SOL005) |
| See note 2 | (Identifier)  vnfPkgIds | NsdInfo | Create NSD  Update NSD  Query/read NSD  (see SOL005) |
| NsdmSubscriptionRequest  -> NsdmNotificationsFilter | Subscription about NSD management  (see SOL005) |
| NsdmSubscription  -> NsdmNotificationsFilter |
| See note 2 | (IdentifierInNsd)  healScript | HealNsRequest -> HealNsData | Heal NS  (see SOL005) |
| See note 2 | (IdentifierInNsd)  aspectId | ScaleNsRequest -> ScaleNsData -> ScaleNsByStepsData | Scale NS  (see SOL005) |
| (IdentifierInNsd)  nsScalingAspectId | ScaleNsRequest -> ScaleNsData -> ScaleNsToLevelData -> NsScaleInfo |
| Node template name of type tosca.nodes.nfv.Sap | (IdentifierInNsd)  sapdId | NsInstance -> SapInfo | Query NS  (see SOL005) |
| InstantiateNsRequest -> SapData | Instantiate NS  (see SOL005) |
| UpdateNsRequest -> SapData | Update NS  (see SOL005) |
| NsLcmOpOcc -> AffectedSap | Query/read information about NS LCM operation occurrence  (see SOL005) |
| NsLcmOperationOccurrenceNotification -> AffectedSap | Notification about NS LCM  (see SOL005) |
| NOTE 1: The entry "SOL005" refers to ETSI GS NFV-SOL 005 [i.10].  NOTE 2: The corresponding TOSCA construct is not included in the present document, the mapping may be updated in future version of the present document. | | | | |

Annex E (informative):  
TOSCA Imperative workflows

# E.1 Purpose

This annex specifies TOSCA Imperative workflows for the NSD and the VNFD by using the grammar defined in TOSCA Simple Profile-YAML-v1.3 [20].

# E.2 TOSCA Imperative workflows for the NSD

## E.2.1 Introduction

TOSCA Imperative workflows based on TOSCA-Simple-Profile-YAML-v1.3 [20] may be used by the NFVO to fulfil the NS LCM operations described in ETSI GS NFV-IFA 013 [i.8]. TOSCA Imperative workflows provide an additional method for implementation of LCM operations in the Nslcm interface defined in clause 7.7.1.1 of the present document.

NOTE: Even if TOSCA Imperative workflows is described in the NSD, the NFVO will still process the NSD with Nslcm operations as defined in clause 7.7.1.1. Since this is an additional method for implementation of NS LCM operations, execution of workflows instead of NS LCM operations is optional and up to the NFVO implementation.

TOSCA Imperative workflows defined in the NSD describe procedures for the NFVO to manage the lifecycle of network services.

Workflows are comprised of steps associated with the NS LCM operations and additional steps that are preamble and postamble to the execution of the former steps. The name of the preamble and postamble steps is constructed according to the following pattern:

< NS\_LCM\_base\_operation\_workflow\_name>\_start\_<step\_name> for preamble steps

< NS\_LCM\_base\_operation\_workflow\_name>\_end\_<step\_name> for postamble steps

Preamble steps are specified before the execution of workflow steps. Postamble steps are specified after the execution of workflow steps.

External and internal stimuli described in clause 7.7.1.4 of the present document, are mapped to workflows as below:

* External stimuli are mapped to TOSCA Imperative workflows, i.e. <NS\_LCM\_base\_operation\_workflow\_name>
* Internal stimuli are mapped to preamble and postamble steps of the workflow

## E.2.2 Definition of an NS workflow

The syntax of TOSCA Imperative workflows for LCM operations on the NS has the following definition:

|  |
| --- |
| workflows:  description: TOSCA Imperative workflows corresponding to NS LCM operations defined in ETSI GS NFV‑IFA 013.  instantiate:  description: This workflow is invoked upon receipt of an Instantiate NS request  # inputs:  steps:  instantiate\_start\_<step\_name>: # Invoked before steps for instantiate LCM operation  # steps for instantiate workflow  instantiate\_end\_<step\_name> : # Invoked after steps for instantiate LCM operation  terminate:  description: This workflow is invoked upon receipt of Terminate NS request  # inputs:  steps:  terminate\_start\_<step\_name> : # Invoked before steps for terminate LCM operation  # steps for terminate workflow  terminate\_end\_<step\_name>: # Invoked after steps for terminate LCM operation |

## E.2.3 Examples

The following example template fragment, based on clause A.8 of the present document, illustrates the use of TOSCA Imperative workflows for NS LCM operations.

When the NFVO executes TOSCA Imperative workflows in the NSD, it uses standard APIs for LCM operations defined in the Or-Vnfm interface and delegates the task to VNFM; The VNFM in turn executes corresponding TOSCA Operations on the VNF, as explained in clause 6.7.1 of the present document.

NOTE 1: The NSD consumer makes available all parameters from the message invoking the NS base LCM operation as inputs to the corresponding TOSCA workflows. The additional parameters for NS base LCM operations are defined as workflow inputs.

NOTE 2: It is out of scope of the present document to specify mapping of SOL003/SOL005 API execution results with the success and failure of workflows.

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3

description: NS TOSCA Imperative Workflows

imports:

- etsi\_nfv\_sol001\_nsd\_types.yaml # all of NSD related TOSCA types as defined in ETSI GS NFV-SOL 001

data\_types:

MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters:

derived\_from: tosca.datatypes.nfv.NsOperationAdditionalParameters

properties:

parameter\_1:

type: string

required: true

default: value\_1

parameter\_2:

type: string

required: true

default: value\_2

node\_types:

tosca.example\_NS:

derived\_from: tosca.nodes.nfv.NS

properties:

descriptor\_id:

type: string

constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]

default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177

designer:

type: string

constraints: [ equal: MyCompany ]

default: MyCompany

name:

type: string

constraints: [ equal: ExampleService ]

default: ExampleService

version:

type: string

constraints: [ equal: '1.0' ]

default: '1.0'

invariant\_id:

type: string

constraints: [ equal: 1111-2222-aaaa-bbbb ]

default: 1111-2222-aaaa-bbbb

flavour\_id:

type: string

constraints: [ equal: 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:

# This abstract node template enables the NSD author to use Nslcm scripts if he does #not use workflows.

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: tosca.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

# Additional parameters input to be defined in the VNFD of VNF\_1.

interfaces:

Vnflcm:

instantiate: . . .

terminate: . . .

VNF\_2:

type: tosca.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

- dependency: VNF\_1

# Additional parameters input to be defined in the VNFD of VNF\_2.

interfaces:

Vnflcm:

instantiate: . . .

terminate: . . .

NsVirtualLink\_1:

type: tosca.nodes.nfv.NsVirtualLink

properties:

connectivity\_type:

layer\_protocols: [ipv4]

flow\_pattern: mesh

vl\_profile:

max\_bitrate\_requirements:

root: 1000

min\_bitrate\_requirements:

root: 1000

workflows:

instantiate: #instantiate workflow

inputs:

additional\_parameters:

type: tosca.datatypes.nfv.NsInstantiateAdditionalParameters

required: false

steps:

# preamble steps for instantiate operation. These correspond to preparatory #steps internal to the NFVO before instantiate operation

instantiate\_start\_step\_1:

. . .

on\_success:

create\_VNF\_1

create\_VNF\_2

# steps for instantiate workflow

create\_VNF\_1: # Step: Instantiate VNF\_1

target: VNF\_1

activities:

call\_operation: tosca.interfaces.nfv.Vnflcm.instantiate

# invoking Vnflcm.instantiate operation enables NFVO to use internal implementation of #Vnflcm.instantiate operation which results in an ETSI GS NFV-SOL 003 API call towards #the VNFM to call VnfInstantiate operation. This enables VNFM to execute LCM operations #to deploy VNF\_1

on\_success:

instantiate\_end\_step\_1

. . .

create\_VNF\_2: # Step: Instantiate VNF\_2

target: VNF\_2

activities:

call\_operation: tosca.interfaces.nfv.Vnflcm.instantiate

# invoking Vnflcm.instantiate operation enables NFVO to use internal implementation of #Vnflcm.instantiate operation which results in an ETSI GS NFV-SOL 003 API call towards #the VNFM to call VnfInstantiate operation. This enables VNFM to execute LCM operations #to deploy VNF\_2

on\_success:

instantiate\_end\_step\_1

. . .

# postamble steps for instantiate operation. These correspond to closing steps #internal to the NFVO after instantiate operation.

instantiate\_end\_step\_1:

. . .

terminate: #terminate workflow

steps:

# preamble steps for terminate operation. These correspond to preparatory #steps internal to the NFVO before terminate operation.

terminate\_start\_step\_1:

. . .

on\_success:

terminate\_VNF\_1

terminate\_VNF\_2

# steps for terminate workflow

terminate\_VNF\_1: # Step: Terminate VNF\_1

target: VNF\_1

activities:

call\_operation: tosca.interfaces.nfv.Vnflcm.terminate

# invoking Vnflcm.terminate operation enables NFVO to use internal implementation of #Vnflcm.terminate operation which results in an ETSI GS NFV-SOL 003 API call towards #the VNFM to call VnfTerminate operation. This enables VNFM to execute LCM operations #to terminate VNF\_1

on\_success:

terminate\_end\_step\_1

. . .

terminate\_VNF\_2: # Step: Terminate VNF\_2

target: VNF\_2

activities:

call\_operation: tosca.interfaces.nfv.Vnflcm.terminate

# invoking Vnflcm.terminate operation enables NFVO to use internal implementation of #Vnflcm.terminate operation which results in an ETSI GS NFV-SOL 003 API call towards #the VNFM to call VnfTerminate operation. This enables VNFM to execute LCM operations #to terminate VNF\_2

on\_success:

terminate\_end\_step\_1

. . .

# post amble steps for terminate operation. These correspond to closing steps #internal to the NFVO after terminate operation.

terminate\_end\_step\_1:

. . .

NOTE 3: As the on\_success keyword is not used between steps inside the workflow for NS LCM base operation, the order of execution is decided by the NFVO.

NOTE 4: As the on\_failure keyword is not present inside the workflow for NS LCM base operation, the error handling is decided by the NFVO.

Annex F (informative):  
Non-Backward Compatible Changes in the present document

# F.1 Introduction

This annex provides the list of non-backward compatible changes during the development of the present document.

# F.2 Non-Backward Compatible changes between version 2.7.1 and 2.6.1

Table F.2-1 provides a list of non-backward compatible changes between version 2.7.1 and version 2.6.1 of the present document.

Table F.2-1: Non-backward compatible changes

|  |  |  |
| --- | --- | --- |
| No. | Description | Clause |
| 1 | nfvi\_constraints type changed from list of string to map of string. | 6.8.3 |
| 2 | The definition of SecurityGroupRule is changed in version 2.7.1, a new policy type tosca.policies.nfv.Abstract.SecurityGroupRule is introduced in the definition, which SecurityGroupRule policy is derived from. | 6.10.13 |
| 3 | PlacementGroup used to be applied for both VNFD and NSD in version 2.6.1, but in version 2.7.1, it is only applied to VNFD. | 6.9 |
| 4 | AffinityRule, AntiAffinityRule used to be applied for both VNFD, NSD in version 2.6.1, but in version 2.7.1, it is only applied to VNFD. | 6.10.10 |
| 5 | SecurityGroupRule policy type used to be applied for VNFD, PNFD and NSD in version 2.6.1, but in version 2.7.1, it is only applied to VNFD. | 6.10.13 |
| 6 | boot\_order type changed from list of string to Boolean. | 6.8.3 |
| 7 | boot\_data type changed from string to BootData data type | 6.8.3 |

Annex G (informative):  
Change History

| Date | Version | Information about changes |
| --- | --- | --- |
| 2016.05 | 0.0.1 | Implemented NFVSOL(16)000005r1\_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 *editHelp*  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 |
| 2018.03 | 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 |
| 2018.03 | 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 |
| 2018.04 | 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 |
| 2018.05 | 0.6.3 | Implemented NFVSOL(18)000168, NFVSOL(18)000169r2, NFVSOL(18)000173, NFVSOL(18)000174r2 |
| 2018.05 | 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 |
| 2018.06 | 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 |
| 2018.06 | 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. |
| 2018.08 | 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 |
| 2018.09 | 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 |
| 2019.03 | 2.5.2 | Implemented NFVSOL(19)000063r4, NFVSOL(19)000068r1, NFVSOL(19)000069r2, NFVSOL(19)000070r1, NFVSOL(19)00080, NFVSOL(19)000107r1, NFVSOL(19)000120r1, NFVSOL(19)000121, NFVSOL(19)00039r5, NFVSOL(19)00087r2, NFVSOL(19)00067r5, NFVSOL(19)00077, NFVSOL(19)00082r3, NFVSOL(19)00085, NFVSOL(19)00086, NFVSOL(19)000106r1 |
| 2019.03 | 2.5.3 | Implemented NFVSOL(19)000084r4, NFVSOL(19)0000101r2, NFVSOL(19)0000165, NFVSOL(19)0000166, NFVSOL(19)0000167r1, NFVSOL(19)0000170r4, NFVSOL(19)0000173, NFVSOL(19)0000163, NFVSOL(19)0000119 |
| 2019.03.22 | 2.5.4 | Editorial modification made by ETSI Secretariat allowing to structure the SOL repository on ETSI Forge in a future proof and maintainable way:   * Forge structure updated: "v2.6.1" tag created. * Yaml filenames updated: version numbers removed from filenames (still included in file header). * Import statements updated: version number removed from imported filenames. * Draft updated: * Updated all references to yaml files * Updated the forge URLs * In B.1: removed the sentence explaining the meaning of x\_y\_z\_ in the filename structure |
| 2019.04.23 | 2.5.5 | 2 comments were raised during the Remote Consensus approval: both requesting to implement the WG SOL approved Change Request in NFVSOL(19)000229r3 onto the final SOL001 draft (see these 2 comments in the RC report).  The present version implements NFVSOL(19)000229r3: adding machine readable meta info inside the yaml file indicating the SOL001 release version to which they apply + other editorial changes. |
| 2019.05 | 2.6.2 | Implemented NFVSOL(19)000162r1, NFVSOL(19)000194r2, NFVSOL(19)000222r2, NFVSOL(19)000241r2, NFVSOL(19)000242r2 |
| 2019.06 | 2.6.3 | Implemented NFVSOL(19)000160r9, NFVSOL(19)000239r3, NFVSOL(19)000248, NFVSOL(19)000262r2, NFVSOL(19)000263r3, NFVSOL(19)000268r3, NFVSOL(19)000269, NFVSOL(19)000270r3, NFVSOL(19)000303r1, NFVSOL(19)000307, NFVSOL(19)000279, NFVSOL(19)000280, NFVSOL(19)000296r3, NFVSOL(19)000301r1, NFVSOL(19)000305r1, NFVSOL(19)000338, NFVSOL(19)000340, NFVSOL(19)000342r1, NFVSOL(19)000344r2, NFVSOL(19)000345r4, NFVSOL(19)000266r7 |
| 2019.08 | 2.6.4 | Implemented NFVSOL(19)000325r8, NFVSOL(19)000346r6, NFVSOL(19)000347r2, NFVSOL(19)000380r1, NFVSOL(19)000383r1, NFVSOL(19)000384r1, NFVSOL(19)000385, NFVSOL(19)000389r1, NFVSOL(19)000428r2, NFVSOL(19)000449 |
| 2019.09 | 2.6.5 | Implemented NFVSOL(19)000386r8, NFVSOL(19)000408r4, NFVSOL(19)000451r8, NFVSOL(19)000559 |
| 2019.11 | 2.6.6 | Implemented NFVSOL(19)000727r1\_SOL001Ed271\_Miscellaneous\_corrections, NFVSOL(19)000642r1\_SOL001ed271\_Annex\_Mapping\_table\_for\_SOL\_API\_NSD\_related\_cons, NFVSOL(19)000645r3\_SOL001ed271\_NBWC\_issue\_list\_annex, NFVSOL(19)000700r1\_SOL001ed271\_updating\_mapping\_table\_in\_A\_9,  NFVSOL(19)000577r2\_SOL001Ed271\_-\_Standards\_Configurable\_Properties,  NFVSOL(19)000593\_SOL001ed271\_adding\_introduction\_in\_clasue\_9\_1, NFVSOL(19)000595\_SOL001ed271\_resolving\_requirement\_occurannce\_issue, NFVSOL(19)000597r3\_SOL001ed271\_deployment\_flavour\_related\_CSAR\_design\_for\_NSD, NFVSOL(19)000598r2\_SOL001ed271\_TOSCA\_Imperative\_workflows\_NSD\_Editor\_s\_Notes, NFVSOL(19)000599r2\_SOL001ed271\_TOSCA\_Imperative\_workflows\_Example\_Editors\_Notes, NFVSOL(19)000607r2\_SOL001ed271\_VNFFG\_clause\_6\_8\_2\_6\_Editor\_s\_note\_handling, NFVSOL(19)000608r2\_SOL001ed271\_VNFFG\_clause\_7\_8\_2\_6\_Editor\_s\_note\_handling, NFVSOL(19)000609r2\_SOL001ed271\_VNFFG\_clause\_7\_8\_5\_1\_Editor\_s\_notes\_handling, NFVSOL(19)000610\_SOL001ed271\_VNFFG\_clause\_7\_8\_6\_1\_Editor\_s\_note\_handling, NFVSOL(19)000611r2\_SOL001ed271\_adding\_TOSCA-Simple-Profile-yaml-v1\_3\_reference, NFVSOL(19)000616\_SOL001ed271\_ConnectivityType, NFVSOL(19)000617r3\_SOL001ed271\_nfviConstraint, NFVSOL(19)000633\_SOL001ed271\_VnfIndicators\_editor\_s\_notes\_resolution, NFVSOL(19)000636r2\_SOL001ed271\_NS\_workflow\_defintion\_update, NFVSOL(19)000702\_SOL001ed271\_adding\_reference\_for\_SwImageData, NFVSOL(19)000703r1\_SOL001ed271\_forwarding\_behaviour\_input\_parameters, NFVSOL(19)000704r1\_SOL001ed271\_correction\_on\_NfpPositionElement\_definition, NFVSOL(19)000705r1\_SOL001ed271\_correction\_on\_descriptor\_id, NFVSOL(19)000741r1\_SOL001ed271\_NfpPositionElement\_node\_type\_improvement, NFVSOL(19)000596r3\_SOL001ed271\_deployment\_flavour\_related\_CSAR\_design\_for\_VNFD, NFVSOL(19)000725r2\_SOL001ed271\_vnfd\_common\_yaml\_file\_for\_v1\_3, NFVSOL(19)000744r2\_SOL001ed271\_Clause\_7\_1\_Introduction\_improvement, NFVSOL(19)000566r9\_SOL001ed271\_adding\_boot\_data\_type, NFVSOL(19)000674r1\_SOL001Ed271\_Monitoring\_Parameters\_-\_Alignement\_with\_IFA027 |
| 2019.11 | 2.6.7 | Implemented NFVSOL(19)000780r3\_SOL001ed271\_\_VnfIndicator\_id\_Issue\_1, NFVSOL(19)000784r1\_SOL001ed271\_solving\_remaining\_editor\_s\_notes, NFVSOL(19)000785r1\_SOL001ed271\_solving\_remaining\_monitoring\_issues, NFVSOL(19)000789\_SOL001ed271\_adding\_boot\_data\_in\_NBWC\_list\_table, NFVSOL(19)000719r12\_SOL001ed271\_forwarding\_capability\_for\_VNF\_node\_type |
| 2020.04 | 2.7.2 | Implemented NFVSOL(20)000238r1\_SOL001ed281\_adding\_NS\_DF\_design\_principle\_in\_annex, NFVSOL(20)000268r2\_SOL001ed281\_VNF\_node\_type\_definitions  rapporteur changes: editorial, changing "will" to "with" in annex C.2 and C.3. |
| 2020.05 | 2.7.3 | Implemented NFVSOL(20)000243\_SOL001ed281\_release\_2\_mirror\_updating\_annex\_C, NFVSOL(20)000263\_SOL001ed281\_Rel-2\_mirror\_VNF\_specific\_datatypes\_naming\_rules, NFVSOL(20)000326\_SOL001ed281\_corrections\_of\_specific\_node\_tpye, NFVSOL(20)000334\_SOL001ed281\_Rel-2\_mirror\_VNF\_specific\_types\_naming\_rules, NFVSOL(20)000375r2\_SOL001ed281\_PNFD\_geographic\_coordinates\_support, NFVSOL(20)000377r1\_SOL001ed281\_add\_VNF\_related\_type\_names\_in\_clause\_6, NFVSOL(20)000379\_SOL001ed281\_add\_NS\_related\_type\_names\_in\_clause\_7, NFVSOL(20)000381\_SOL001ed281\_clarification\_of\_using\_VnfMonitoringParameter, NFVSOL(20)000383\_SOL001ed281\_clarification\_on\_ip\_address\_type, NFVSOL(20)000385r1\_SOL001ed281\_clarification\_on\_VirtualLinkable\_capability, NFVSOL(20)000193r7\_SOL001ed281\_example\_corrections |
| 2020.06 | 2.7.4 | Implemented NFVSOL(20)000493\_SOL001ed281\_NS\_specific\_node\_type\_file, NFVSOL(20)000516r1\_SOL001ed281\_Fixing\_optional\_properties\_with\_defaults, NFVSOL(20)000539\_SOL001ed281\_Rel-2\_mirror\_of\_267\_Ignoring\_imported\_topology |

# History

|  |  |  |
| --- | --- | --- |
| **Document history** | | |
| V2.5.1 | December 2018 | Publication |
| V2.6.1 | May 2019 | Publication |
| V2.7.1 | January 2020 | Publication |
| V2.8.1 | September 2020 | Publication |
|  |  |  |