Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; NFV descriptors based on TOSCA specification
## Contents

Intellectual Property Rights .................................................................................................................. 24
Foreword .................................................................................................................................................. 24
Modal verbs terminology ......................................................................................................................... 24

1 Scope .................................................................................................................................................. 25
2 References ......................................................................................................................................... 25
2.1 Normative references ..................................................................................................................... 25
2.2 Informative references .................................................................................................................... 26
3 Definition of terms, symbols and abbreviations ................................................................................. 28
3.1 Terms .............................................................................................................................................. 28
3.2 Symbols .......................................................................................................................................... 28
3.3 Abbreviations ................................................................................................................................. 28
4 Overview of TOSCA model ................................................................................................................. 28
5 General concept of using TOSCA to model NFV descriptors ............................................................ 28
5.1 Introduction ................................................................................................................................. 28
5.2 Network Service Descriptor .......................................................................................................... 29
5.3 Virtualised Network Function Descriptor ..................................................................................... 29
5.4 Physical Network Function Descriptor ......................................................................................... 30
5.5 tosca_definitions_version and Namespace prefix ..................................................................... 30
5.6 Imports statement ......................................................................................................................... 30
5.6.1 VNFD TOSCA service template ................................................................................................. 30
5.6.2 NSD TOSCA service template ................................................................................................. 31
5.6.3 PNFD TOSCA service template ............................................................................................... 31
5.7 Type extension ............................................................................................................................... 32
5.7.1 Introduction .............................................................................................................................. 32
5.7.2 Rules ......................................................................................................................................... 32
5.7.3 VNFD Types ............................................................................................................................... 33
5.7.4 NSD types ................................................................................................................................ 34
5.7.5 Security-sensitive properties in extended data types ................................................................. 34
5.8 Non-Backward Compatible changes ............................................................................................... 35
5.9 Use of TOSCA functions ................................................................................................................. 35

6 VNFD TOSCA model ............................................................................................................................ 36
6.1 Introduction ..................................................................................................................................... 36
6.2 Data Types ..................................................................................................................................... 38
6.2.1 tosca.datatypes.nfv.CpProtocolData ....................................................................................... 38
6.2.1.1 Description .......................................................................................................................... 38
6.2.2 tosca.datatypes.nfv.AddressData ............................................................................................ 38
6.2.2.1 Description .......................................................................................................................... 38
6.2.3 tosca.datatypes.nfv.L2AddressData .......................................................................................... 38
6.2.3.1 Description .......................................................................................................................... 38
6.2.4 tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements .................................................. 38
6.2.4.1 Description .......................................................................................................................... 38
6.2.4.2 Properties ............................................................................................................................ 38
6.2.4.3 Definition ............................................................................................................................. 39
6.2.4.4 Examples ............................................................................................................................. 39
6.2.4.5 Additional Requirements .................................................................................................... 39
6.2.5 tosca.datatypes.nfv.L3AddressData .......................................................................................... 40
6.2.5.1 Description .......................................................................................................................... 40
6.2.5.2 Properties ............................................................................................................................ 40
6.2.5.3 Definition ............................................................................................................................. 41
6.2.5.4 Examples ............................................................................................................................. 41
6.2.5.5 Additional Requirements .................................................................................................... 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.27.5</td>
<td>Additional Requirements ........................................................................70</td>
</tr>
<tr>
<td>6.2.28</td>
<td>tosca.datatypes.nfv.ScalingAspect ..................................................70</td>
</tr>
<tr>
<td>6.2.28.1</td>
<td>Description ........................................................................................70</td>
</tr>
<tr>
<td>6.2.28.2</td>
<td>Properties ..........................................................................................70</td>
</tr>
<tr>
<td>6.2.28.3</td>
<td>Definition .......................................................................................70</td>
</tr>
<tr>
<td>6.2.28.4</td>
<td>Examples ............................................................................................71</td>
</tr>
<tr>
<td>6.2.28.5</td>
<td>Additional Requirements ........................................................................71</td>
</tr>
<tr>
<td>6.2.29</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements .....................................71</td>
</tr>
<tr>
<td>6.2.29.1</td>
<td>Description ........................................................................................71</td>
</tr>
<tr>
<td>6.2.30</td>
<td>tosca.datatypes.nfv.ConnectivityType ................................................71</td>
</tr>
<tr>
<td>6.2.31</td>
<td>tosca.datatypes.nfv.VnfConfigurableProperties ...................................71</td>
</tr>
<tr>
<td>6.2.31.1</td>
<td>Description ........................................................................................71</td>
</tr>
<tr>
<td>6.2.31.2</td>
<td>Properties ..........................................................................................72</td>
</tr>
<tr>
<td>6.2.31.3</td>
<td>Definition .......................................................................................73</td>
</tr>
<tr>
<td>6.2.31.4</td>
<td>Examples ............................................................................................74</td>
</tr>
<tr>
<td>6.2.31.5</td>
<td>Additional Requirements ........................................................................75</td>
</tr>
<tr>
<td>6.2.32</td>
<td>tosca.datatypes.nfv.VnfAdditionalConfigurableProperties .......................75</td>
</tr>
<tr>
<td>6.2.32.1</td>
<td>Description ........................................................................................75</td>
</tr>
<tr>
<td>6.2.32.2</td>
<td>Properties ..........................................................................................75</td>
</tr>
<tr>
<td>6.2.32.3</td>
<td>Definition .......................................................................................75</td>
</tr>
<tr>
<td>6.2.32.4</td>
<td>Examples ............................................................................................76</td>
</tr>
<tr>
<td>6.2.32.5</td>
<td>Additional Requirements ........................................................................76</td>
</tr>
<tr>
<td>6.2.33</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributes ....................................76</td>
</tr>
<tr>
<td>6.2.33.1</td>
<td>Description ........................................................................................76</td>
</tr>
<tr>
<td>6.2.33.2</td>
<td>Properties ..........................................................................................76</td>
</tr>
<tr>
<td>6.2.33.3</td>
<td>Definition .......................................................................................76</td>
</tr>
<tr>
<td>6.2.33.4</td>
<td>Examples ............................................................................................77</td>
</tr>
<tr>
<td>6.2.33.5</td>
<td>Additional Requirements ........................................................................77</td>
</tr>
<tr>
<td>6.2.34</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions .....................77</td>
</tr>
<tr>
<td>6.2.34.1</td>
<td>Description ........................................................................................77</td>
</tr>
<tr>
<td>6.2.34.2</td>
<td>Properties ..........................................................................................77</td>
</tr>
<tr>
<td>6.2.34.3</td>
<td>Definition .......................................................................................77</td>
</tr>
<tr>
<td>6.2.34.4</td>
<td>Examples ............................................................................................78</td>
</tr>
<tr>
<td>6.2.34.5</td>
<td>Additional Requirements ........................................................................78</td>
</tr>
<tr>
<td>6.2.35</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata ........................78</td>
</tr>
<tr>
<td>6.2.35.1</td>
<td>Description ........................................................................................78</td>
</tr>
<tr>
<td>6.2.35.2</td>
<td>Properties ..........................................................................................78</td>
</tr>
<tr>
<td>6.2.35.3</td>
<td>Definition .......................................................................................78</td>
</tr>
<tr>
<td>6.2.35.4</td>
<td>Examples ............................................................................................78</td>
</tr>
<tr>
<td>6.2.35.5</td>
<td>Additional Requirements ........................................................................79</td>
</tr>
<tr>
<td>6.2.36</td>
<td>tosca.datatypes.nfv.Qos ........................................................................79</td>
</tr>
<tr>
<td>6.2.36.1</td>
<td>Description ........................................................................................79</td>
</tr>
<tr>
<td>6.2.37</td>
<td>tosca.datatypes.nfv.LogicalNodeData ...................................................79</td>
</tr>
<tr>
<td>6.2.37.1</td>
<td>Description ........................................................................................79</td>
</tr>
<tr>
<td>6.2.37.2</td>
<td>Properties ..........................................................................................79</td>
</tr>
<tr>
<td>6.2.37.3</td>
<td>Definition .......................................................................................80</td>
</tr>
<tr>
<td>6.2.37.4</td>
<td>Examples ............................................................................................80</td>
</tr>
<tr>
<td>6.2.37.5</td>
<td>Additional Requirements ........................................................................80</td>
</tr>
<tr>
<td>6.2.38</td>
<td>tosca.datatypes.nfv.SwImageData ..........................................................80</td>
</tr>
<tr>
<td>6.2.38.1</td>
<td>Description ........................................................................................80</td>
</tr>
<tr>
<td>6.2.38.2</td>
<td>Properties ..........................................................................................80</td>
</tr>
<tr>
<td>6.2.38.3</td>
<td>Definition .......................................................................................81</td>
</tr>
<tr>
<td>6.2.38.4</td>
<td>Examples ............................................................................................82</td>
</tr>
<tr>
<td>6.2.38.5</td>
<td>Additional Requirements ........................................................................82</td>
</tr>
<tr>
<td>6.2.39</td>
<td>tosca.datatypes.nfv.VirtualBlockStorageData .........................................83</td>
</tr>
<tr>
<td>6.2.39.1</td>
<td>Description ........................................................................................83</td>
</tr>
<tr>
<td>6.2.39.2</td>
<td>Properties ..........................................................................................83</td>
</tr>
<tr>
<td>6.2.39.3</td>
<td>Definition .......................................................................................83</td>
</tr>
<tr>
<td>6.2.39.4</td>
<td>Examples ............................................................................................84</td>
</tr>
<tr>
<td>6.2.39.5</td>
<td>Additional Requirements ........................................................................84</td>
</tr>
<tr>
<td>6.2.40</td>
<td>tosca.datatypes.nfv.VirtualObjectStorageData ..........................................84</td>
</tr>
<tr>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>6.2.51.2</td>
<td>Properties</td>
</tr>
<tr>
<td>6.2.51.3</td>
<td>Definition</td>
</tr>
<tr>
<td>6.2.51.4</td>
<td>Examples</td>
</tr>
<tr>
<td>6.2.51.5</td>
<td>Additional Requirements</td>
</tr>
<tr>
<td>6.2.52</td>
<td>tosca.datatypes.nfv.VnfProfile</td>
</tr>
<tr>
<td>6.2.53</td>
<td>tosca.datatypes.nfv.ChecksumData</td>
</tr>
<tr>
<td>6.2.54</td>
<td>tosca.datatypes.nfv.BootDataVimSpecificProperties</td>
</tr>
<tr>
<td>6.2.55</td>
<td>tosca.datatypes.nfv.BootData</td>
</tr>
<tr>
<td>6.2.56</td>
<td>tosca.datatypes.nfv.KvpData</td>
</tr>
<tr>
<td>6.2.57</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
<tr>
<td>6.2.58</td>
<td>tosca.datatypes.nfv.ContentOrFileData</td>
</tr>
<tr>
<td>6.2.59</td>
<td>tosca.datatypes.nfv.BootDataVimSpecificProperties</td>
</tr>
<tr>
<td>6.2.60</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
<tr>
<td>6.2.61</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
<tr>
<td>6.2.62</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
<tr>
<td>6.2.63</td>
<td>tosca.datatypes.nfv.VnfChangeCurrentPackageOperationConfiguration</td>
</tr>
</tbody>
</table>

ETSI GS NFV-SOL 001 V4.2.1 (2022-01)
6.2.62.2 Properties .............................................................................................................................. 108
6.2.62.3 Definition ............................................................................................................................... 108
6.2.62.4 Examples ............................................................................................................................... 109
6.2.62.5 Additional Requirements ...................................................................................................... 109
6.2.63 tosca.datatypes.nfv.VnfCreateSnapshotOperationConfiguration ........................................... 109
6.2.63.1 Description ............................................................................................................................ 109
6.2.63.2 Properties ............................................................................................................................ 109
6.2.63.3 Definition ............................................................................................................................ 109
6.2.63.4 Examples ............................................................................................................................ 109
6.2.63.5 Additional Requirements ...................................................................................................... 109
6.2.64 tosca.datatypes.nfv.VnfRevertToSnapshotOperationConfiguration .......................................... 109
6.2.64.1 Description ............................................................................................................................ 109
6.2.64.2 Properties ............................................................................................................................ 110
6.2.64.3 Definition ............................................................................................................................ 110
6.2.64.4 Examples ............................................................................................................................ 110
6.2.64.5 Additional Requirements ...................................................................................................... 110
6.2.65 tosca.datatypes.nfv.ServicePortData ..................................................................................... 110
6.2.65.1 Description ............................................................................................................................ 110
6.2.65.2 Properties ............................................................................................................................ 110
6.2.65.3 Definition ............................................................................................................................ 111
6.2.65.4 Examples ............................................................................................................................ 111
6.2.65.5 Additional Requirements ...................................................................................................... 111
6.2.66 tosca.datatypes.nfv.AdditionalServiceData ............................................................................. 112
6.2.66.1 Description ............................................................................................................................ 112
6.2.66.2 Properties ............................................................................................................................ 112
6.2.66.3 Definition ............................................................................................................................ 112
6.2.66.4 Examples ............................................................................................................................ 112
6.2.66.5 Additional Requirements ...................................................................................................... 112
6.2.67 tosca.datatypes.nfv.VnfLcmOpCoord ..................................................................................... 113
6.2.67.1 Description ............................................................................................................................ 113
6.2.67.2 Properties ............................................................................................................................ 113
6.2.67.3 Definition ............................................................................................................................ 115
6.2.67.4 Examples ............................................................................................................................ 115
6.2.67.5 Additional Requirements ...................................................................................................... 115
6.2.68 tosca.datatypes.nfv.InputOpCoordParams ............................................................................. 116
6.2.68.1 Description ............................................................................................................................ 116
6.2.68.2 Properties ............................................................................................................................ 116
6.2.68.3 Definition ............................................................................................................................ 116
6.2.69 tosca.datatypes.nfv.OutputOpCoordParams ........................................................................... 116
6.2.69.1 Description ............................................................................................................................ 116
6.2.69.2 Properties ............................................................................................................................ 116
6.2.69.3 Definition ............................................................................................................................ 116
6.2.70 tosca.datatypes.nfv.ExtendedResourceData ......................................................................... 117
6.2.70.1 Description ............................................................................................................................ 117
6.2.70.2 Properties ............................................................................................................................ 117
6.2.70.3 Definition ............................................................................................................................ 117
6.2.70.4 Examples ............................................................................................................................ 118
6.2.70.5 Additional Requirements ...................................................................................................... 118
6.2.71 tosca.datatypes.nfv.Hugepages ............................................................................................ 118
6.2.71.1 Description ............................................................................................................................ 118
6.2.71.2 Properties ............................................................................................................................ 118
6.2.71.3 Definition ............................................................................................................................ 119
6.2.71.4 Examples ............................................................................................................................ 119
6.2.71.5 Additional Requirements ...................................................................................................... 119
6.2.72 tosca.datatypes.nfv.MaxNumberOfImpactedInstances ........................................................... 119
6.2.72.1 Description ............................................................................................................................ 119
6.2.72.2 Properties ............................................................................................................................ 120
6.2.72.3 Definition ............................................................................................................................ 120
6.2.72.4 Examples ............................................................................................................................ 120
6.2.72.5 Additional Requirements ...................................................................................................... 120
6.2.73 tosca.datatypes.nfv.MinNumberOfPreservedInstances ............................................................ 121
6.2.73.1 Description ............................................................................................................................ 121
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8.4.2</td>
<td>Properties</td>
<td>133</td>
</tr>
<tr>
<td>6.8.4.3</td>
<td>Definition</td>
<td>133</td>
</tr>
<tr>
<td>6.8.5</td>
<td>tosca.relationships.nfv.DeploysTo</td>
<td>134</td>
</tr>
<tr>
<td>6.8.5.1</td>
<td>Description</td>
<td>134</td>
</tr>
<tr>
<td>6.8.5.2</td>
<td>Properties</td>
<td>134</td>
</tr>
<tr>
<td>6.8.5.3</td>
<td>Definition</td>
<td>134</td>
</tr>
<tr>
<td>6.8.6</td>
<td>tosca.relationships.nfv.MciopAssociates</td>
<td>134</td>
</tr>
<tr>
<td>6.8.6.1</td>
<td>Description</td>
<td>134</td>
</tr>
<tr>
<td>6.8.6.2</td>
<td>Properties</td>
<td>134</td>
</tr>
<tr>
<td>6.8.6.3</td>
<td>Definition</td>
<td>134</td>
</tr>
<tr>
<td>6.8.7</td>
<td>Interface Types</td>
<td>135</td>
</tr>
<tr>
<td>6.8.7.1</td>
<td>tosca.interfaces.nfv.Vnflcm</td>
<td>135</td>
</tr>
<tr>
<td>6.8.7.1.1</td>
<td>Description</td>
<td>135</td>
</tr>
<tr>
<td>6.8.7.1.2</td>
<td>Definition</td>
<td>136</td>
</tr>
<tr>
<td>6.8.7.1.3</td>
<td>Additional Requirements</td>
<td>139</td>
</tr>
<tr>
<td>6.8.7.1.4</td>
<td>Support of LCM scripts</td>
<td>140</td>
</tr>
<tr>
<td>6.8.7.1.5</td>
<td>Examples</td>
<td>140</td>
</tr>
<tr>
<td>6.8.7.2</td>
<td>tosca.interfaces.nfv.Vnflcm</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.2.1</td>
<td>Description</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.2.2</td>
<td>Definition</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.2.3</td>
<td>Examples</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.3</td>
<td>tosca.interfaces.nfv.ChangeCurrentVnffPackage</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.3.1</td>
<td>Description</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.3.2</td>
<td>Definition</td>
<td>143</td>
</tr>
<tr>
<td>6.8.7.3.3</td>
<td>Examples</td>
<td>144</td>
</tr>
<tr>
<td>6.8.8</td>
<td>Node Types</td>
<td>144</td>
</tr>
<tr>
<td>6.8.8.1</td>
<td>tosca.nodes.nfv.VNF</td>
<td>144</td>
</tr>
<tr>
<td>6.8.8.1.1</td>
<td>Description</td>
<td>144</td>
</tr>
<tr>
<td>6.8.8.1.2</td>
<td>Properties</td>
<td>144</td>
</tr>
<tr>
<td>6.8.8.1.3</td>
<td>Attributes</td>
<td>146</td>
</tr>
<tr>
<td>6.8.8.1.4</td>
<td>Requirements</td>
<td>147</td>
</tr>
<tr>
<td>6.8.8.1.5</td>
<td>Capabilities</td>
<td>147</td>
</tr>
<tr>
<td>6.8.8.1.6</td>
<td>Definition</td>
<td>147</td>
</tr>
<tr>
<td>6.8.8.1.7</td>
<td>Artifact</td>
<td>149</td>
</tr>
<tr>
<td>6.8.8.1.8</td>
<td>Additional Requirements</td>
<td>149</td>
</tr>
<tr>
<td>6.8.8.1.9</td>
<td>Example</td>
<td>150</td>
</tr>
<tr>
<td>6.8.8.2</td>
<td>tosca.nodes.nfv.VnffExtCp</td>
<td>153</td>
</tr>
<tr>
<td>6.8.8.2.1</td>
<td>Description</td>
<td>153</td>
</tr>
<tr>
<td>6.8.8.2.2</td>
<td>Properties</td>
<td>153</td>
</tr>
<tr>
<td>6.8.8.2.3</td>
<td>Attributes</td>
<td>153</td>
</tr>
<tr>
<td>6.8.8.2.4</td>
<td>Requirements</td>
<td>153</td>
</tr>
<tr>
<td>6.8.8.2.5</td>
<td>Capabilities</td>
<td>154</td>
</tr>
<tr>
<td>6.8.8.2.6</td>
<td>Definition</td>
<td>154</td>
</tr>
<tr>
<td>6.8.8.2.7</td>
<td>Additional Requirements</td>
<td>154</td>
</tr>
<tr>
<td>6.8.8.2.8</td>
<td>Example</td>
<td>154</td>
</tr>
<tr>
<td>6.8.8.3</td>
<td>tosca.nodes.nfv.VnffCompute</td>
<td>155</td>
</tr>
<tr>
<td>6.8.8.3.1</td>
<td>Description</td>
<td>155</td>
</tr>
<tr>
<td>6.8.8.3.2</td>
<td>Properties</td>
<td>155</td>
</tr>
<tr>
<td>6.8.8.3.3</td>
<td>Attributes</td>
<td>156</td>
</tr>
<tr>
<td>6.8.8.3.4</td>
<td>Requirements</td>
<td>156</td>
</tr>
<tr>
<td>6.8.8.3.5</td>
<td>Capabilities</td>
<td>157</td>
</tr>
<tr>
<td>6.8.8.3.6</td>
<td>Definition</td>
<td>157</td>
</tr>
<tr>
<td>6.8.8.3.7</td>
<td>Additional requirements</td>
<td>158</td>
</tr>
<tr>
<td>6.8.8.3.8</td>
<td>Example</td>
<td>159</td>
</tr>
<tr>
<td>6.8.8.4</td>
<td>tosca.nodes.nfv.VnffVirtualBlockStorage</td>
<td>164</td>
</tr>
<tr>
<td>6.8.8.4.1</td>
<td>Description</td>
<td>164</td>
</tr>
<tr>
<td>6.8.8.4.2</td>
<td>Properties</td>
<td>164</td>
</tr>
<tr>
<td>6.8.8.4.3</td>
<td>Attributes</td>
<td>165</td>
</tr>
<tr>
<td>6.8.8.4.4</td>
<td>Requirements</td>
<td>165</td>
</tr>
<tr>
<td>6.8.8.4.5</td>
<td>Capabilities</td>
<td>165</td>
</tr>
<tr>
<td>6.8.8.4.6</td>
<td>Definition</td>
<td>165</td>
</tr>
<tr>
<td>6.8.8.4.7</td>
<td>Additional requirements</td>
<td>165</td>
</tr>
</tbody>
</table>
6.1.10.9.2 Properties ................................................................................................................. 195
6.1.10.9.3 Definition .................................................................................................................. 195
6.1.10.9.4 Examples .................................................................................................................. 195
6.1.10.10 AffinityRule, AntiAffinityRule.................................................................................. 195
6.1.10.11 Description .................................................................................................................. 195
6.1.10.12 Properties .................................................................................................................. 196
6.1.10.13 targets ......................................................................................................................... 197
6.1.10.14 Definition .................................................................................................................. 197
6.1.10.15 Examples .................................................................................................................... 198
6.1.10.16 tosca.policies.nfv.Abstract.SecurityGroupRule......................................................... 198
6.1.10.17 Description .................................................................................................................. 198
6.1.10.18 tosca.policies.nfv.SupportedVnfProfile ...................................................................... 199
6.1.10.19 Description .................................................................................................................. 199
6.1.10.20 Properties .................................................................................................................. 199
6.1.10.21 targets ......................................................................................................................... 200
6.1.10.22 Definition .................................................................................................................. 200
6.1.10.23 Additional requirements .......................................................................................... 200
6.1.10.24 Example ...................................................................................................................... 200
6.1.10.25 tosca.policies.nfv.SecurityGroupRule ........................................................................... 200
6.1.10.26 Description .................................................................................................................. 200
6.1.10.27 Properties .................................................................................................................. 200
6.1.10.28 targets ......................................................................................................................... 200
6.1.10.29 Definition .................................................................................................................. 200
6.1.10.30 Additional Requirements .......................................................................................... 200
6.1.10.31 Example ...................................................................................................................... 200
6.1.10.32 tosca.policies.nfv.VnfIndicator .................................................................................... 201
6.1.10.33 Description .................................................................................................................. 201
6.1.10.34 Properties .................................................................................................................. 201
6.1.10.35 Definition .................................................................................................................. 201
6.1.10.36 Additional requirements .......................................................................................... 201
6.1.10.37 Example ...................................................................................................................... 201
6.1.10.38 tosca.policies.nfv.VnfPackageChange .......................................................................... 202
6.1.10.39 Description .................................................................................................................. 202
6.1.10.40 Properties .................................................................................................................. 202
6.1.10.41 Definition .................................................................................................................. 202
6.1.10.42 Additional Requirements .......................................................................................... 202
6.1.10.43 Example ...................................................................................................................... 202
6.1.10.44 tosca.policies.nfv.LcmCoordinationAction ................................................................. 214
6.1.10.45 Description .................................................................................................................. 214
6.1.10.46 Properties .................................................................................................................. 214
6.1.10.47 Definition .................................................................................................................. 214
6.1.10.48 Additional Requirements .......................................................................................... 215
6.1.10.49 Example ...................................................................................................................... 215
6.1.10.50 tosca.policies.nfv.LcmCoordinationsForLcmOperation.............................................. 215
6.1.10.51 Description .................................................................................................................. 215
6.1.10.52 Properties .................................................................................................................. 215
6.1.10.53 Definition .................................................................................................................. 215
6.1.10.54 Additional Requirements .......................................................................................... 215
6.1.10.55 Example ...................................................................................................................... 215
6.1.10.56 VNFD TOSCA service template design ..................................................................... 216
6.1.10.57 General ...................................................................................................................... 216
6.1.10.58 Single or multiple deployment flavour design with two levels of service templates .... 216
6.1.10.59 Single deployment flavour design with one service template .................................... 217
6.1.10.60 Package change (handling the Change current VNF Package request) ...................... 218
6.1.10.61 NSD TOSCA model .................................................................................................. 219
6.1.10.62 Introduction .................................................................................................................. 219
7.1.1 Data Types ....................................................................................................................... 219
7.1.2 Void ................................................................................................................................. 220
7.1.3 tosca.datatypes.nfv.VnfProfile ...................................................................................... 220
7.1.4 Description ....................................................................................................................... 220
7.1.5 Properties ....................................................................................................................... 220
7.1.6 Definition ....................................................................................................................... 221
7.1.7 Examples ......................................................................................................................... 221
7.1.8 tosca.datatype.nfv.NsVfProfile ....................................................................................... 222
7.1.9 Description ....................................................................................................................... 222
7.1.10 Properties ...................................................................................................................... 222
7.1.11 Definition ....................................................................................................................... 222
7.1.12 Examples ....................................................................................................................... 222
ANNEX A (INFORMATIVE): Examples

A.1 Deployment flavour design mapping ..............................................................310
A.1.1 Introduction ........................................................................................................310
A.1.2 Design principle for VNF deployment flavour .............................................310
A.1.3 Design principle for NS deployment flavour .........................................................310
A.2 VNFD with deployment flavour modelling design example ........................311
A.3 VNF external connection point ........................................................................321
A.3.1 General .................................................................................................................321
A.3.2 External connection point re-exposing an internal connection point........321
A.3.3 External connection point connected to an internal virtual link ................325
A.4 VNFD modelling design example by using TOSCA composition ..................326
A.5 VNFD with Single deployment flavour modelling design example ..............331
A.6 Scaling and Instantiation Level examples .........................................................335
A.6.1 ScalingAspect and InstantiationLevels policies with uniform delta ..........335
A.6.2 ScalingAspect and InstantiationLevels policies with non-uniform deltas ....341
A.7 Service Access Point .........................................................................................345
A.7.1 General .................................................................................................................345
A.7.2 VNF External connection point exposing as a SAP ........................................346
A.7.3 SAP connected to an NS virtual link .................................................................348
A.8 NSD with Single deployment flavour modelling design example ...............349
A.9 Mapping between NFV IM and TOSCA concepts ........................................353
A.9.1 Introduction ........................................................................................................353
A.9.2 Mapping between ETSI GS NFV-IFA 011 IM and TOSCA concepts ..........353
A.9.3 Mapping between ETSI GS NFV-IFA 014 IM and TOSCA concepts .........355
A.10 PNFD modelling design example .................................................................356
A.11 NSD with Multiple deployment flavour modelling design example ...........358
A.12 NSD with nested NS design example .........................................................364
A.13 Virtual IP address connection point ............................................................369
A.14 NSD VNF Forwarding Graph design example ...........................................376
A.15 Auto-scale and auto-heal design ................................................................386
A.15.1 Introduction ....................................................................................................386
A.15.2 Auto-scale and auto-heal design with use of VNF indicator in VNFD .........386
A.15.3 Auto-scale design with use of VNF indicator in NSD .................................390
A.16 VDU connection point in trunk mode .........................................................397
A.17 NS scaling ....................................................................................................400
A.18 VNFD illustrating OsContainer modeling example .................................................................405
Annex B (normative): etsi_nfv_sol001_type definitions ..............................................................411
  B.1 Purpose ......................................................................................................................................411
  B.2 VNFD type definitions file ........................................................................................................411
  B.3 NSD type definitions file ..........................................................................................................411
  B.4 PNFD type definitions file ........................................................................................................412
  B.5 Common type definitions file ..................................................................................................412
Annex C (normative): Conformance ..............................................................................................413
  C.1 Purpose ......................................................................................................................................413
  C.2 NFV TOSCA YAML service template ......................................................................................413
  C.3 NFV TOSCA processor .............................................................................................................414
Annex D (informative): Mapping between properties of TOSCA types and API attributes ......415
  D.1 Introduction .............................................................................................................................415
  D.2 VNFD-related constructs ........................................................................................................415
  D.3 NSD-related constructs ............................................................................................................424
Annex E (informative): TOSCA Imperative workflows .................................................................429
  E.1 Purpose ......................................................................................................................................429
  E.2 TOSCA Imperative workflows for the NSD .............................................................................429
  E.2.1 Introduction ...........................................................................................................................429
  E.2.2 Definition of an NS workflow ...............................................................................................429
  E.2.3 Examples ...............................................................................................................................430
Annex F (informative): Non-Backward Compatible Changes in the GS .......................................435
  F.1 Introduction ...............................................................................................................................435
  F.2 Non-Backward Compatible changes between version 2.7.1 and 2.6.1 .........................................435
  F.3 Non-Backward Compatible changes between version 2.8.1 and 3.3.1 .........................................435
  F.4 Non-Backward Compatible changes between version 3.3.1 and 3.5.1 .........................................436
Annex G (informative): Change History .........................................................................................437
History ................................................................................................................................................444
Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are 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 Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, 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.

**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.

Foreword

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

Modal verbs terminology

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

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

The present document specifies a data model for NFV descriptors, using the TOSCA-Simple-Profile-YAML-v1.3 [20], 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-YAML-v1.3 [20].

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.


[5] Private Enterprise Numbers registry at IANA.

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


[10] ISO 3166 (all parts): "Codes for the representation of names of countries and their subdivisions".


[14] Hash Function Textual Names registry at IANA.

NOTE: Available at https://www.iana.org/assignments/hash-function-text-names.
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) Release 4; 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] Void.

[i.4] Void.

[i.5] Void.

[i.6] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 4; 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.
ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

ETSI GS NFV-SOL 003: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

ETSI GS NFV-SOL 005: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

ETSI GS NFV-SOL 007: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; Network Service Descriptor File Structure Specification".

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.

Openstack Metadata service.

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

Openstack User-data service.

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

Openstack Personality service.


ETSI NFV registry of VimConnectionInfo information.

NOTE: Available at http://register.etsi.org/NFV.

IETF RFC 4090: "Fast Reroute Extensions to RSVP-TE for LSP Tunnels".

TOSCA-Simple-Profile-yaml-v1.2: "TOSCA Simple Profile in YAML Version 1.2".

ETSI GS NFV-SOL 001 (V3.3.1): "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification".

ETSI GS NFV-SOL 001 (V2.8.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

ETSI GS NFV-SOL 001 (V3.5.1): "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification".

ETSI GS NFV-SOL 001 (V2.6.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

ETSI GS NFV-SOL 001 (V2.7.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".
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-v1.3 [20].

**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-v1.3 [20], 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-v1.3 [20].

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).

When processing TOSCA service templates modelling all or part of an NFV descriptor, the consumer of the NFV descriptor shall comply with and implement the semantics of any of the keynames defined in clause 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20] and used in clauses 5, 6, 7 and 8 of the present document. The presence of other keynames defined in clause 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20] shall not cause the NFV descriptor to be rejected and the consumer of the NFV descriptor may choose to ignore them and their associated contents, unless they are part of the type definitions files referred to in Annex B of the present document.
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). A VDU describes the resources needed to deploy and manage the lifecycle of a VNFC. A VDU includes internal Connection Point Descriptors (CPDs) that describe internal connection points that can either be used to connect a VNFC to an internal virtual link or be re-exposed outside the VNFC 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.
The information within a VNFD is structured according to one or more VNF deployment flavours (VnDf) 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-v1.3 [20] with the associated Namespace Alias value defined in the TOSCA-Simple-Profile-YAML-v1.3 [20].

NOTE 1: This implies that service templates complying to the present document can only import service templates that reference the same version - and thus use the same grammar - as the importing service template. This is a restriction compared to the TOSCA-Simple-Profile-YAML-v1.3 specification [20].

NOTE 2: As specified in TOSCA-Simple-Profile-YAML-v1.3 [20], the grammar used in TOSCA-Simple-Profile-YAML-v1.2 [i.18] is still supported with deprecation. The present document indicates the cases in which support with deprecation is applicable.

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>
<thead>
<tr>
<th>Namespace Prefix</th>
<th>Specification Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>toscanfv</td>
<td>The TOSCA namespace prefix of all the TOSCA types as specified in the present document.</td>
</tr>
</tbody>
</table>

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-v1.3 [20], 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:

```yaml
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-v1.3 [20], 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:

```yaml
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:

```yaml
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-v1.3 [20], 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:

```yaml
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:

```yaml
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 (e.g. additional parameters to LCM operations).

5.7.2 Rules

Type extension may be applied to NFV types defined in the present document within the limits specified in table 5.7.3-1 and table 5.7.4-1, 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 in table 5.7.3-1 and table 5.7.4-1.

Existing properties may be extended according to the following rules:

a) A scalar property shall not be extended to another type (e.g. a string property shall not be replaced with an integer property or with a complex property of the same name).

b) A complex property of data type "X" may only be extended to a property of type "Y" where "Y" is derived from "X" according to the present rules (recursive rule: present rules applied to each property of the derived data type).

c) A property of type list with entry schema "X" may only be extended to a list with entry schema "Y" where "Y" is an extension of "X" according to the present rules (recursive rule: present rules applied to the elements of the list).

d) A property of type map with entry schema "X" may only be extended to a map with entry schema "Y" where "Y" is an extension of "X" according to the present rules (recursive rule: present rules applied to the values of the map).

In general, the above rules apply to introducing/extending other elements beyond properties such as capabilities, requirements, interfaces, operations, inputs, etc. as well.
5.7.3 VNFD Types

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Keyname</th>
<th>Property name</th>
</tr>
</thead>
<tbody>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>properties</td>
<td>modifiable_attributes (as a new property)</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>requirements</td>
<td>configurable_properties (as a new property). See note 4.</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>interfaces</td>
<td>New requirements with capability type</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>interfaces</td>
<td>VirtualLinkable (as new requirements).</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>interfaces</td>
<td>Vnf lcm.{operation_name}.inputs.additional_parameters (as a new property)</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>interfaces</td>
<td>Vnf lcm.{operation_name}.inputs (as new properties).</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>interfaces</td>
<td>VnfIndicator.notifications (as new notifications). See note 4.</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VnfIndicator</td>
<td>attributes</td>
<td>One attribute of primitive type per VNF indicator may be added.</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VnfIndicator</td>
<td>attributes</td>
<td>One attribute of type integer per scaling aspect may be added. It holds the value of the current scale level.</td>
</tr>
<tr>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributes</td>
<td>properties</td>
<td>extensions (as a new property)</td>
</tr>
<tr>
<td>tosca.interfaces.nfv.VnfIndicator</td>
<td>notifications</td>
<td>one notification may be added per Vnf indicator. See note 3.</td>
</tr>
<tr>
<td>tosca.interfaces.nfv.ChangeCurrentVnfPackage</td>
<td>operations</td>
<td>{change_current_package_script} (as new operation).</td>
</tr>
<tr>
<td>tosca.interfaces.nfv.ChangeCurrentVnfPackage</td>
<td>operations</td>
<td>{change_current_package_script}.inputs.additional_parameters (as new property).</td>
</tr>
<tr>
<td>tosca.policies.nfv.LcmCoordinationAction</td>
<td>policies</td>
<td>See clause 6.10.16.4</td>
</tr>
</tbody>
</table>
NOTE 1: 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 2: 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 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>
<thead>
<tr>
<th>Type</th>
<th>Keyname</th>
<th>Property name</th>
</tr>
</thead>
<tbody>
<tr>
<td>tosca.nodes.nfv.NS</td>
<td>requirements</td>
<td>New requirements with capability type VirtualLinkable (as new requirements).</td>
</tr>
<tr>
<td></td>
<td>interfaces</td>
<td>Nslcm.{operation_name}.inputs.additional_ parameters (as a new property).</td>
</tr>
<tr>
<td>tosca.policies.nfv.NsAutoScale</td>
<td>policies</td>
<td>See clause 7.10.15.</td>
</tr>
</tbody>
</table>

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.

### 5.7.5 Security-sensitive properties in extended data types

The definition of the properties of some of the data types derived from the data types specified in the present document may include the following metadata:

```
sensitive: "true"
```

NOTE: Double quotes are needed to avoid that the parser interprets it as the Boolean value true.

This metadata indicates that the property holds security-sensitive information (e.g. passwords).

It is out of the scope of the present document to specify the exact behaviour of a functional block handling security-sensitive properties. The intent of this metadata is to signal not to expose the value of the property by means such as user interfaces, logging files, programmatic interfaces, etc.

Specific handling of these properties when they are used as parameters in the APIs is defined in the affected specifications, e.g. ETSI GS NFV-SOL 003 [i.9].

Extension of the types listed in table 5.7.5-1 may include properties with this metadata.
Table 5.7.5-1: VNFD and NSD extensible data types

<table>
<thead>
<tr>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions</td>
</tr>
<tr>
<td>tosca.datatypes.nfv.VnfAdditionalConfigurableProperties</td>
</tr>
<tr>
<td>tosca.datatypes.nfv.VnfOperationAdditionalParameters</td>
</tr>
<tr>
<td>tosca.datatypes.nfv.NsOperationAdditionalParameters</td>
</tr>
</tbody>
</table>

In this version of the present document the use of the security-sensitive tagging is limited to properties defined in the extended types. The use of the security-sensitive tagging in properties defined in the present document is not supported.

The security-sensitive tagging is foreseen for properties whose values are expected to be dynamically set from the APIs. Tagging a property that has a value assigned in the VNFD as security-sensitive does not prevent its exposure when the complete VNFD is exposed. Therefore, assigning a value in the VNFD should be avoided.

5.8 Non-Backward Compatible changes

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

5.9 Use of TOSCA functions

The TOSCA service templates complying with the present document may use the TOSCA functions listed in table 5.9-1. Use of these TOSCA functions shall comply with the provisions in section 4 of TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 5.9-1: Supported TOSCA functions

<table>
<thead>
<tr>
<th>TOSCA function</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_property</td>
</tr>
<tr>
<td>get_artifact</td>
</tr>
<tr>
<td>get_input (see note 1)</td>
</tr>
<tr>
<td>get_attribute (see note 2)</td>
</tr>
<tr>
<td>TOSCA intrinsic functions (see note 3)</td>
</tr>
</tbody>
</table>

NOTE 1: The get_input function is used to retrieve the values of parameters declared in the input section of a service template and assign them properties. Service templates complying with the present document may only use the get_input function to assigning values to the properties listed in table 5.9-2.

NOTE 2: Service templates complying with the present document may only use the get_attribute function to retrieve the value of the following attributes:

a) In a VNF node template: scale_status attribute.

b) In an NS node template: scale_status attribute and any VNF node attribute holding the value of a VNF indicator.

NOTE 3: TOSCA intrinsic functions (concat, join and token) are defined in section 4.3 of TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 5.9-2: Applicable properties for get_input

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>flavour_id</td>
</tr>
<tr>
<td></td>
<td>modifiable_attributes</td>
</tr>
<tr>
<td></td>
<td>configurable_properties</td>
</tr>
<tr>
<td>tosca.nodes.nfv.Vdu.Compute</td>
<td>configurable_properties</td>
</tr>
</tbody>
</table>

NOTE: Input values are either assigned in the service templates or received from the APIs. The mapping between TOSCA properties and API attributes is described in Annex D 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-Profile-YAML-v1.3 [20] from which they are derived from. The full definition of all types can be found in the following clauses.

NOTE 1: The autoScale rule with use of VNF monitoring parameters specified in ETSI GS NFV-IFA 011 [1] is not supported in this version of the present document.

NOTE 2: The monitoring parameters for Oscontainer based VNF 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

<table>
<thead>
<tr>
<th>ETSI GS NFV-IFA 011 [1] Elements</th>
<th>VNFD TOSCA types</th>
<th>Derived from</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNFD</td>
<td>tosca.nodes.nfv.VNF</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>Vdu (Vdu)</td>
<td>n/a (see note 1)</td>
<td>n/a</td>
</tr>
<tr>
<td>Cpd (Connection Point)</td>
<td>tosca.nodes.nfv.Cp</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>VduCpd (internal connection point)</td>
<td>tosca.nodes.nfv.VduCp</td>
<td>tosca.nodes.nfv.Cp</td>
</tr>
<tr>
<td>VipCpd</td>
<td>tosca.nodes.nfv.VipCp</td>
<td>tosca.nodes.nfv.Cp</td>
</tr>
<tr>
<td>VnfVirtualLinkDesc (Virtual Link)</td>
<td>tosca.nodes.nfv.VnfVirtualLink</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>VnfExtCpd (External Connection Point)</td>
<td>tosca.nodes.nfv.VnfExtCp</td>
<td>tosca.nodes.nfv.Cp</td>
</tr>
<tr>
<td>VnfDf</td>
<td>n/a (see note 2)</td>
<td>n/a</td>
</tr>
<tr>
<td>SwimageDesc</td>
<td>tosca.artifacts.nfv.Swimage</td>
<td>tosca.artifacts_DeploymentImage</td>
</tr>
<tr>
<td>OsContainerDesc</td>
<td>tosca.nodes.nfv.Vdu.OsContainer</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>VnfConfigurableProperties</td>
<td>tosca.datatypes.nfv.VnfConfigurableProperties</td>
<td>tosca.datatypes.Root</td>
</tr>
<tr>
<td>VnffinfoModifiableAttributes</td>
<td>tosca.datatypes.nfv.VnffinfoModifiableAttributes</td>
<td>tosca.datatypes.Root</td>
</tr>
<tr>
<td>MciopProfile</td>
<td>n/a (see note 3)</td>
<td>n/a</td>
</tr>
</tbody>
</table>


NOTE 2: The VnfDf information element is represented as a TOSCA service template.

NOTE 3: There is not a direct mapping between the MciopProfile in ETSI GS NFV-IFA 011 [1] and the tosca.nodes.nfv.Mciop. However, the 'deploymentOrder' and 'associatedVdu' attributes defined in the MciopProfile element are mapped to tosca.nodes.nfv.Mciop. The 'affinityOrAntiAffinityGroupId' is mapped to tosca.policies.nfv.AffinityRule or tosca.policies.nfv.AntiAffinityRule.
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 for VNFs when all its virtualisation containers are realized as VMs. 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 when all the virtualisation containers of the VNF are realized as VMs

Figure 6.1-2 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 when all the virtualisation containers of the VNF are realized as OsContainers.

Figure 6.1-2: Service template VNFD overview when all the virtualisation containers of the VNF are realized as OsContainers
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualNetworkInterfaceRequirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VirtualNetworkInterfaceRequirements</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provides a human readable name for the requirement.</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provides a human readable description of the requirement.</td>
</tr>
<tr>
<td>support_mandatory</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>Indicates whether fulfilling the constraint is mandatory (TRUE) for successful operation or desirable (FALSE).</td>
</tr>
<tr>
<td>network_interface_requirements</td>
<td>yes</td>
<td>map of string</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>More information regarding the usage of this property is available at:</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-------------------------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>nic_io_requirements</td>
<td>no</td>
<td>tosca.datatypes.nfv.LogicalNodeData</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

### 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>RequestedAdditionalCapability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:RequestedAdditionalCapability</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.RequestedAdditionalCapability</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requested_additional_capability_name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies a requested additional capability for the VDU.</td>
</tr>
<tr>
<td>support_mandatory</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>Indicates whether the requested additional capability is mandatory for successful operation.</td>
</tr>
<tr>
<td>min_requested_additional_capability_version</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the minimum version of the requested additional capability.</td>
</tr>
<tr>
<td>preferred_requested_additional_capability_version</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the preferred version of the requested additional capability.</td>
</tr>
<tr>
<td>target_performance_parameters</td>
<td>yes</td>
<td>map of string</td>
<td></td>
<td>Identifies specific attributes, dependent on the requested additional capability type.</td>
</tr>
</tbody>
</table>
6.2.6.3 Definition

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

```yaml
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: true
    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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualMemory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VirtualMemory</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualMemory</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_mem_size</td>
<td>yes</td>
<td>scalar-unit.size</td>
<td></td>
<td>Amount of virtual memory.</td>
</tr>
<tr>
<td>virtual_mem_oversubscr</td>
<td>no</td>
<td>string</td>
<td></td>
<td>The memory core oversubscription policy in terms of virtual memory to physical memory on the platform.</td>
</tr>
<tr>
<td>vdu_mem_requirements</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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: <a href="https://register.etsi.org">https://register.etsi.org</a>.</td>
</tr>
<tr>
<td>numa_enabled</td>
<td>yes</td>
<td>boolean</td>
<td>default: false</td>
<td>It specifies the memory allocation to be cognisant of the relevant process/core allocation.</td>
</tr>
</tbody>
</table>

6.2.7.3 Definitions

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

```python
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualCpu</td>
<td>cpu_architecture</td>
<td>no</td>
<td>string</td>
<td></td>
<td>CPU architecture type. Examples are x86, ARM.</td>
</tr>
<tr>
<td>tosca:VirtualCpu</td>
<td>num_virtual_cpu</td>
<td>yes</td>
<td>integer</td>
<td>greater_than: 0</td>
<td>Number of virtual CPUs.</td>
</tr>
<tr>
<td>tosca:VirtualCpu</td>
<td>virtual_cpu_clock</td>
<td>no</td>
<td>scalar-unit.frequency</td>
<td>Minimum virtual CPU clock rate.</td>
<td></td>
</tr>
<tr>
<td>tosca:VirtualCpu</td>
<td>virtual_cpu_oversubscription_policy</td>
<td>no</td>
<td>string</td>
<td></td>
<td>CPU core oversubscription policy e.g. the relation of virtual CPU cores to physical CPU cores/threads.</td>
</tr>
<tr>
<td>tosca:VirtualCpu</td>
<td>vdu_cpu_requirements</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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: <a href="https://register.etsi.org">https://register.etsi.org</a>.</td>
</tr>
<tr>
<td>tosca:VirtualCpu</td>
<td>virtual_cpu_pinning</td>
<td>no</td>
<td>tosca.datatypes.nfv.VirtualCpuPinning</td>
<td>The virtual CPU pinning configuration for the virtualised compute resource.</td>
<td></td>
</tr>
</tbody>
</table>

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.

6.2.8.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualCpuPinning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VirtualCpuPinning</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualCpuPinning</td>
</tr>
</tbody>
</table>

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.
### 6.2.9.3 Definition

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

```plaintext
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-v1.3 [20].

---

**Table 6.2.9.2-1: Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_cpu_pinning_policy</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Indicates the policy for CPU pinning. The policy can take values of &quot;static&quot; or &quot;dynamic&quot;. In case of &quot;dynamic&quot; 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 &quot;static&quot; the allocation is requested to be according to the virtual_cpu_pinning_rule.</td>
</tr>
<tr>
<td>virtual_cpu_pinning_rule</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>Provides the list of rules for allocating virtual CPU cores to logical CPU cores/threads.</td>
</tr>
</tbody>
</table>
Table 6.2.10.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfcConfigurableProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfcConfigurableProperties</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfcConfigurableProperties</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additional_vnfc_configurable_properties</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfcAdditionConfigurableProperties</td>
<td></td>
<td>Describes additional configuration for VNFC that can be modified using the ModifyVnfInfo operation.</td>
</tr>
</tbody>
</table>

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

Example definition of configurable properties without properties assignment value.

tosca_definitions_version: tosca_simple_yaml_1_3

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:
      ...

Example definition of configurable properties with properties assignment value.

tosca_definitions_version: tosca_simple_yaml_1_3

... node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ...
    interfaces:
      Vnflcm:
        type: tosca.interfaces.nfv.Vnflcm

  MyCompany.nodes.nfv.VduAux:
    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.

6.2.10.5 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfcAdditionalConfigurableProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfcAdditionalConfigurableProperties</td>
</tr>
</tbody>
</table>

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.

6.2.11.4 Examples
See clause 6.2.10.4.

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 (for VM based VDU) or Vdu.OsContainerDeployableUnit (for Oscontainer based VDU) 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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduProfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VduProfile</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VduProfile</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_number_of_instances</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Minimum number of instances of the VNFC based on this Vdu.Compute (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU) that is permitted to exist for a particular VNF deployment flavour.</td>
</tr>
<tr>
<td>max_number_of_instances</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Maximum number of instances of the VNFC based on this Vdu.Compute (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU) that is permitted to exist for a particular VNF deployment flavour.</td>
</tr>
<tr>
<td>nfvi_maintenance_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.NfviMaintenanceInfo</td>
<td></td>
<td>Provides information on the impact tolerance and rules to be observed when instance(s) of the Vdu.Compute (for VM based VDU) are impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades). See notes 2 and 3.</td>
</tr>
</tbody>
</table>

NOTE 1: 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 (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU).

NOTE 2: The impact tolerance and rules also apply to the VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes connected to the Vdu.Compute (for VM based VDU) via one particular occurrence of the virtual_storage requirement and to each VduCp node connected to the Vdu.Compute (for VM based VDU) via one particular occurrence of the virtual_binding capability.

NOTE 3: An NFVI level operation (e.g. restart of a virtual machine) can impact a VNF and the VNF may be able to tolerate only a limited number of such impacts simultaneously. The nfvi_maintenance_info provides constraints related to detection and tolerance so that negative impact on VNF functionality can be avoided during NFVI maintenance operations.
6.2.12.3 Definition

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

```yaml
tosca.datatypes.nfv.VduProfile:
    derived_from: tosca.datatypes.Root
    description: describes additional instantiation data for a given Vdu.Compute (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU) 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 (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU) 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 (for VM based VDU) or Vdu.OsContainerDeployableUnit node (for Oscontainer based VDU) that is permitted to exist for a particular VNF deployment flavour.
            required: true
            constraints:
                - greater_or_equal: 0
        nfvi_maintenance_info:
            type: tosca.datatypes.nfv.NfviMaintenanceInfo
            description: Provides information on the impact tolerance and rules to be observed when instance(s) of the Vdu.Compute (for VM based VDU) are impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).
            required: false
```
6.2.12.4 Examples

tosca_definitions_version: tosca_simple_yaml_1_3

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
      # per_vnfc_instance property not present or set to true
    capabilities:
      virtual_storage

  VirtualStorage_A2:
    type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
    properties:
      # omitted for brevity
      # per_vnfc_instance property not present or set to true
    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:

- If the 'per_vnfc_instance' property of the VirtualBlockStorage, VirtualObjectStorage or VirtualFileStorage nodes connected to the Vdu.Compute node is set to 'true' or absent: 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.

  NOTE: If 'per_vnfc_instance' property is set to 'false' only one instance of the storage node shall exist.

- 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VlProfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VlProfile</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_bitrate_requirements</td>
<td>yes</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td></td>
<td>Specifies the maximum bitrate requirements for a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>min_bitrate_requirements</td>
<td>yes</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td></td>
<td>Specifies the minimum bitrate requirements for a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>qos</td>
<td>no</td>
<td>tosca.datatypes.nfv.Qos</td>
<td></td>
<td>Specifies the QoS requirements of a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>virtual_link_protocol_data</td>
<td>no</td>
<td>list of tосsa.datatypes.nfv.VirtualLinkProtocolData</td>
<td></td>
<td>Specifies the protocol data for a virtual link. If more than 1 values are present, the order shall be the same as the order of the layer_protocols occurrences in the connectivity_type property of the same VnfVirtualLink node, i.e. the first occurrence of the virtual_link_protocol_data represents the highest layer protocol data, and the last occurrence represents the lowest layer protocol data.</td>
</tr>
</tbody>
</table>

**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:

```yaml
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 \texttt{tosca.datatypes.nfv.VirtualLinkProtocolData}

6.2.14.1 Description

The \texttt{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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualLinkProtocolData</td>
<td>tosca.nfv.VirtualLinkProtocolData</td>
<td>tosca.datatypes.nfv.VirtualLinkProtocolData</td>
</tr>
</tbody>
</table>

6.2.14.2 Properties

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>associated_layer_protocol</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Identifies one of the protocols a virtualLink gives access to (ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire) as specified by the connectivity_type property.</td>
</tr>
<tr>
<td>l2_protocol_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.L2ProtocolData</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>l3_protocol_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.L3ProtocolData</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

6.2.14.3 Definition

The syntax of the \texttt{VirtualLinkProtocolData} data type shall comply with the following definition:

```yaml
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 describes L2 protocol data for a given virtual link used in a specific VNF deployment flavour. Table 6.2.15.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

| Table 6.2.15.1-1: Type name, shorthand, and URI |
|---------------------------------|-----------------|
| Shorthand Name | L2ProtocolData |
| Type Qualified Name | tosca:nfv: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>
<thead>
<tr>
<th>Table 6.2.15.2-1: Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>network_type</td>
</tr>
<tr>
<td>vlan_transparent</td>
</tr>
<tr>
<td>mtu</td>
</tr>
<tr>
<td>segmentation_id</td>
</tr>
</tbody>
</table>

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:

```markdown
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-v1.3 [20].
### 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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the network name associated with this L3 protocol.</td>
</tr>
<tr>
<td>ip_version</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Specifies IP version of this L3 protocol.</td>
</tr>
<tr>
<td>cidr</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Specifies the CIDR (Classless Inter-Domain Routing) of this L3 protocol.</td>
</tr>
<tr>
<td>ip_allocation_pools</td>
<td>no</td>
<td>list of</td>
<td></td>
<td>Specifies the allocation pools with start and end IP addresses for this L3 protocol. The value may be overridden at run-time.</td>
</tr>
<tr>
<td>gateway_ip</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Specifies the gateway IP address for this L3 protocol. The value may be overridden at run-time.</td>
</tr>
<tr>
<td>dhcp_enabled</td>
<td>no</td>
<td>boolean</td>
<td></td>
<td>Indicates whether DHCP (Dynamic Host Configuration Protocol) is enabled or disabled for this L3 protocol. The value may be overridden at run-time.</td>
</tr>
<tr>
<td>ipv6_address_mode</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Specifies IPv6 address mode. May be present when the value of the ipVersion attribute is &quot;ipv6&quot; and shall be absent otherwise. The value may be overridden at run-time.</td>
</tr>
</tbody>
</table>

### 6.2.16.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>IpAllocationPool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.IpAllocationPool</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.IpAllocationPool</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_ip_address</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>The IP address to be used as the first one in a pool of addresses derived from the cidr block full IP range</td>
</tr>
<tr>
<td>end_ip_address</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>The IP address to be used as the last one in a pool of addresses derived from the cidr block full IP range</td>
</tr>
</tbody>
</table>

6.2.17.3 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>InstantiationLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscnfv:InstantiationLevel</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.InstantiationLevel</td>
</tr>
</tbody>
</table>

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.
### 6.2.18.3 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.VduLevel</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VduLevel</td>
</tr>
</tbody>
</table>

#### 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.
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfLcmOperationsConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfLcmOperationsConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfLcmOperationsConfiguration</td>
</tr>
</tbody>
</table>
```

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instantiate</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfInstantiateOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the InstantiateVnf operation.</td>
</tr>
<tr>
<td>scale</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfScaleOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the ScaleVnf operation.</td>
</tr>
<tr>
<td>scale_to_level</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the ScaleVnfToLevel operation.</td>
</tr>
<tr>
<td>change_flavour</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the changeVnfFlavourOpConfig operation.</td>
</tr>
<tr>
<td>heal</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfHealOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the HealVnf operation.</td>
</tr>
<tr>
<td>terminate</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfTerminateOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the TerminateVnf operation.</td>
</tr>
<tr>
<td>operate</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfOperateOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the OperateVnf operation.</td>
</tr>
<tr>
<td>change_ext_connectivity</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the changeExtVnfConnectivityOpConfig operation.</td>
</tr>
<tr>
<td>create_snapshot</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfCreateSnapshotOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the CreateVnfSnapshot operation.</td>
</tr>
<tr>
<td>revert_to_snapshot</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfRevertToSnapshotOperationConfiguration</td>
<td></td>
<td>Configuration parameters for the RevertToVnfSnapshot operation.</td>
</tr>
</tbody>
</table>

6.2.20.3 Definition

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

```yaml
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
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfInstantiateOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfInstantiateOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfInstantiateOperationConfiguration</td>
</tr>
</tbody>
</table>

6.2.21.2 Properties

None.

6.2.21.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfScaleOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfScaleOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfScaleOperationConfiguration</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaling_by_more_than_one_step_supported</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>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. &quot;not supported&quot;.</td>
</tr>
</tbody>
</table>

6.2.22.3 Definition

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

```python
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfScaleToLevelOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfScaleToLevelOperationConfiguration</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary_target_levels_supported</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>Signals whether scaling according to the parameter &quot;scaleInfo&quot; is supported by this VNF.</td>
</tr>
</tbody>
</table>
6.2.23.3 Definition

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

```python
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfHealOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfHealOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfHealOperationConfiguration</td>
</tr>
</tbody>
</table>

Table 6.2.24.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>causes</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>Supported &quot;cause&quot; parameter values.</td>
</tr>
</tbody>
</table>

6.2.24.3 Definition

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

```python
tosca.datatypes.nfv.VnfHealOperationConfiguration:
    derived_from: tosca.datatypes.Root
    description: represents information that affect the invocation of the HealVnf operation
    properties:
        causes:
```

E.T.S.I.
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnfTerminateOperationConfiguration</td>
<td>tosca.datatypes.nfv.VnfTerminateOperationConfiguration</td>
<td>tosca.datatypes.nfv.VnfTerminateOperationConfiguration</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_graceful_termination_timeout</td>
<td>yes</td>
<td>scalar-unit.time</td>
<td></td>
<td>Minimum timeout value for graceful termination of a VNF instance.</td>
</tr>
<tr>
<td>max_recommended_graceful_termination_time</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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
**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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfOperateOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.VnfOperateOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfOperateOperationConfiguration</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_graceful_stop_timeout</td>
<td>yes</td>
<td>scalar-unit.time</td>
<td></td>
<td>Minimum timeout value for graceful stop of a VNF instance.</td>
</tr>
<tr>
<td>max_recommended_graceful_stop_timeout</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**6.2.26.3 Definition**

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScaleInfo</td>
<td>tosca.datatypes.nfv.ScaleInfo</td>
<td>tosca.datatypes.nfv.ScaleInfo</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale_level</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>The scale level for a particular aspect.</td>
</tr>
</tbody>
</table>

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:
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScalingAspect</td>
<td>tosca.datatypes.nfv.ScalingAspect</td>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable name of the aspect.</td>
</tr>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the aspect.</td>
</tr>
<tr>
<td>max_scale_level</td>
<td>yes</td>
<td>integer</td>
<td>positiveInteger</td>
<td>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 that 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.</td>
</tr>
<tr>
<td>step_deltas</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>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).</td>
</tr>
</tbody>
</table>

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
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfConfigurableProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:VnfConfigurableProperties</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfConfigurableProperties</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_autoscale_enabled</td>
<td>no</td>
<td>boolean</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>is_autoheal_enabled</td>
<td>no</td>
<td>boolean</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>vnfm_interface_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>vnfm_oauth_server_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.OauthServerInfo</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vnf_oauth_server_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.OauthServerInfo</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>additional_configurable_property</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfAddConfigurableProperties</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

6.2.31.3 Definition

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

```yaml
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 values.

tosca_definitions_version: tosca_simple_yaml_1_3
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.
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfAdditionalConfigurableProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:VnfAdditionalConfigurableProperties</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfAdditionalConfigurableProperties</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_writable_anytime</td>
<td>yes</td>
<td>boolean</td>
<td>It specifies whether these additional configurable properties are writeable (TRUE) at anytime (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.</td>
<td></td>
</tr>
</tbody>
</table>

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfInfoModifiableAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributes</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributes</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extensions</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions</td>
<td></td>
<td>&quot;Extension&quot; properties of VnfInfo that are writeable.</td>
</tr>
<tr>
<td>metadata</td>
<td>no</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata</td>
<td></td>
<td>&quot;Metadata&quot; properties of VnfInfo that are writeable.</td>
</tr>
</tbody>
</table>

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

```yaml
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

The following example shows an example of a derived VnfInfoModifiableAttributesExtensions data type that contains one security-sensitive property.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3

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:
    password:
      type: string
    metadata:
      sensitive: "true"
```

See clause 6.8.1.9 for other examples.

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfInfoModifiableAttributesExtensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:VnfInfoModifiableAttributesExtensions</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions</td>
</tr>
</tbody>
</table>

6.2.34.2 Properties

None.

6.2.34.3 Definition

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

```yaml
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 \texttt{tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata}

6.2.35.1 Description
The \texttt{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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfInfoModifiableAttributesMetadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata</td>
</tr>
</tbody>
</table>

6.2.35.2 Properties
None.

6.2.35.3 Definition
The syntax of the \texttt{VnfInfoModifiableAttributesMetadata} data type shall comply with the following definition:

\begin{verbatim}
tosca.datatypes.nfv.VnfInfoModifiableAttributesMetadata:
  derived_from: tosca.datatypes.Root
  description: is an empty base type for deriving data types for describing VNF-specific metadata
\end{verbatim}

6.2.35.4 Examples
Example metadata definition:

\begin{verbatim}
tosca_definitions_version: tosca_simple_yam1_1_3
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
  
  mycompany.datatypes.nfv.VnfInfoModifiableAttributes:
    derived_from: tosca.datatypes.nfv.VnfInfoModifiableAttributes
    properties:
      metadata:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesMetadata
\end{verbatim}
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>LogicalNodeData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:LogicalNodeData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.LogicalNodeData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical_node_requirements</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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: <a href="https://register.etsi.org">https://register.etsi.org</a>.</td>
</tr>
</tbody>
</table>
6.2.37.3 Definition

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

```python
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.
```

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>SwImageData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv:SwImageData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.SwImageData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Name of this software image.</td>
</tr>
<tr>
<td>version</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Version of this software image.</td>
</tr>
<tr>
<td>provider</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provider of this software image.</td>
</tr>
<tr>
<td>checksum</td>
<td>yes</td>
<td>tosca.data types.nfv.Checksum Data</td>
<td></td>
<td>Checksum of the software image file.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>container_format</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Valid values: See YAML definition constraints. The container format describes the container file format in which software image is provided.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Description of valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aki: a kernel image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ami: a machine image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ari: a ramdisk image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bare: the image does not have a container or metadata envelope</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>docker: docker container format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ova: OVF package in a tarfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ovf: OVF container format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Future versions of the present document may extend the list of possible values. See note 1.</td>
</tr>
<tr>
<td>disk_format</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Valid values: See YAML definition constraints. The disk format of a software image is the format of the underlying disk image.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Description of valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aki: a kernel image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ami: a machine image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ari: a ramdisk image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>iso: an archive format for the data contents of an optical disc, such as CD-ROM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>qcOW2: a common disk image format, which can expand dynamically and supports copy on write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>raw: an unstructured disk image format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vdi: a common disk image format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vhd: a common disk image format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vHDx: enhanced version of VHD format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vMDk: a common disk image format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Future versions of the present document may extend the list of possible values. See note 2.</td>
</tr>
<tr>
<td>min_disk</td>
<td>yes</td>
<td>scalar-size</td>
<td>greater_or_equal: 0 B</td>
<td>The minimal disk size requirement for this software image.</td>
</tr>
<tr>
<td>min_ram</td>
<td>no</td>
<td>scalar-size</td>
<td></td>
<td>The minimal RAM requirement for this software image.</td>
</tr>
<tr>
<td>size</td>
<td>yes</td>
<td>scalar-size</td>
<td>greater_or_equal: 0 B</td>
<td>The size of this software image.</td>
</tr>
<tr>
<td>operating_system</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the operating system used in the software image.</td>
</tr>
<tr>
<td>supported_virtualisation_environment</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>Identifies the virtualisation environments (e.g. hypervisor) compatible with this software image.</td>
</tr>
</tbody>
</table>

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:

```yaml
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
```

ETSI
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualBlockStorageData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.VirtualBlockStorageData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualBlockStorageData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size_of_storage</td>
<td>yes</td>
<td>scalar-unit.size</td>
<td>greater_or_equal: 0 B</td>
<td>Size of virtualised storage resource.</td>
</tr>
<tr>
<td>vdu_storage_requirements</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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: <a href="https://register.etsi.org">https://register.etsi.org</a>.</td>
</tr>
<tr>
<td>rdma_enabled</td>
<td>yes</td>
<td>boolean</td>
<td>default: false</td>
<td>Indicate if the storage support RDMA.</td>
</tr>
</tbody>
</table>

6.2.39.3  Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualObjectStorageData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VirtualObjectStorageData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualObjectStorageData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_size_of_storage</td>
<td>no</td>
<td>scalar-unit.size</td>
<td>greater_or_equal: 0 B</td>
<td>Maximum size of virtualised storage resource.</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualObjectFileData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:VirtualFileStorageData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualFileStorageData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>size_of_storage</td>
<td>yes</td>
<td>scalar-unit.size</td>
<td>greater_or_equal: 0 B</td>
<td>Size of virtualised storage resource.</td>
</tr>
<tr>
<td>file_system_protocol</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>The shared file system protocol (e.g. NFS, CIFS).</td>
</tr>
</tbody>
</table>

6.2.41.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinkBitrateLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VirtualLinkBitrateLevel</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VirtualLinkBitrateLevel</td>
</tr>
</tbody>
</table>

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

6.2.42.3  Definition

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

```yaml
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-v1.3 [20].
<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfOperationAdditionalParameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfOperationAdditionalParameters</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfOperationAdditionalParameters</td>
</tr>
</tbody>
</table>

### 6.2.43.2 Properties

None.

### 6.2.43.3 Definition

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

```yaml
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

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
node_types:
  MyCompany.nodes.nfv.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ...
    interfaces:
      Vnflcm:
        operations:
          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
```

The following example shows the declaration of an additional parameter to receive a password in the `instantiate` operation. The metadata sensitive: "true" indicates that this property holds security-sensitive information.
tosca_definitions_version: tosca_simple_yaml_1_3

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

data_types:
  MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      inst_password:
        type: string
        required: true
      metadata:
        sensitive: "true"

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfChangeFlavourOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfChangeFlavourOperationConfiguration</td>
</tr>
</tbody>
</table>

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.VnfChangeExtConnectivityOperationConfiguration

6.2.45.1 Description

The VnfChangeExtConnectivityOperationConfiguration data type represents information that affect the invocation of the ChangeExtVnfConnectivity operation, as specified in ETSI GS NFV-IFA 011 [1]. 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfChangeExtConnectivityOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfChangeExtConnectivityOperationConfiguration</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfcMonitoringParameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VnfcMonitoringParameter</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfcMonitoringParameter</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable name of the monitoring parameter.</td>
</tr>
<tr>
<td>performance_metric</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition</td>
<td>Identifies a performance metric to be monitored. Performance metric values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>constraints</td>
<td>shall be either set to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• A corresponding measurement name defined in clause 7.2 of ETSI GS NFV-IFA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>027 [7], without appending a sub-counter. In this case the VNFM computes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>these measurements from lower-level metrics collected from the VIM. See</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>note.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• A corresponding measurement name defined in clause 7.1 of ETSI GS NFV-IFA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>027 [7], without appending a sub-counter. In this case the VNFM collects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>these metrics from the VIM for all compute, storage and network resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>allocated to the VNFC instance.</td>
</tr>
<tr>
<td>collection_period</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>Describes the periodicity at which to collect the performance information.</td>
</tr>
</tbody>
</table>

NOTE: The measured object type for 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 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:

```yaml
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
```
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualLinkMonitoringParameter</td>
<td>tosca.nfv: VirtualLinkMonitoringParameter</td>
<td>tosca.datatypes.nfv.VirtualLinkMonitoringParameter</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable name of the monitoring parameter.</td>
</tr>
</tbody>
</table>
| performance_metric | yes      | string             | Valid values: See YAML definition constraints | 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:

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>InterfaceDetails</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.InterfaceDetails</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.InterfaceDetails</td>
</tr>
</tbody>
</table>

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

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

6.2.49.3 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>UriComponents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscannfv.UriComponents</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.UriComponents</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheme</td>
<td>yes</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the scheme component of a URI, as per IETF RFC 3986 [8]</td>
</tr>
<tr>
<td>authority</td>
<td>no</td>
<td>tosca.datatypes.nfv.UriAuthority</td>
<td></td>
<td>Corresponds to the authority component of a URI, as per IETF RFC 3986 [8]; See note</td>
</tr>
<tr>
<td>path</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the path component of a URI, as per IETF RFC 3986 [8]</td>
</tr>
<tr>
<td>query</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the query component of a URI, as per IETF RFC 3986 [8]</td>
</tr>
<tr>
<td>fragment</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the fragment component of a URI, as per IETF RFC 3986 [8]</td>
</tr>
</tbody>
</table>

**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:

```yaml
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 [22], 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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>UriAuthority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscnfv:UriAuthority</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.UriAuthority</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_info</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>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.</td>
</tr>
<tr>
<td>host</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the host field of the authority component of a URI, as per IETF RFC 3986 [8]. See note 1.</td>
</tr>
<tr>
<td>port</td>
<td>no</td>
<td>string</td>
<td>String values shall comply with IETF RFC 3986 [8]</td>
<td>Corresponds to the port field of the authority component of a URI, as per IETF RFC 3986 [8]. See note 2.</td>
</tr>
</tbody>
</table>

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:

```yaml
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
```
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 [22], the host property and port properties may be included and the user_info property shall not be included. If the host property is included and the value is a registered name, it is assumed that means are in place to resolve the host name to the correct IP address. If the host property is not included, it is assumed that the VNFM will use the IP address associated to one of the connection point instances created from the VnfExpCp and VduCp node types declared as a target of the SupportedVnfInterface policy.

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

### 6.2.52  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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ChecksumData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:ChecksumData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.ChecksumData</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Describes the algorithm used to obtain the checksum value, as described in [14].</td>
</tr>
<tr>
<td>hash</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Contains the result of applying the algorithm indicated by the algorithm property to the data to which this ChecksumData refers.</td>
</tr>
</tbody>
</table>
6.2.53.3 Definition

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

```yaml
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

```yaml
<some_tosca_entity>:
  properties:
    checksum:
      algorithm: sha-256
      hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664d6eda90ed3cd74b4f9d
```

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfmInterfaceInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfmInterfaceInfo</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface_name</td>
<td>yes</td>
<td>string</td>
<td>Valid values</td>
<td>Identifies an interface produced by the VNFM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vnf_lcm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vnf_pm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vnf_fm</td>
<td></td>
</tr>
<tr>
<td>details</td>
<td>no</td>
<td>tosca.datatypes.nfv.InterfaceDetails</td>
<td></td>
<td>Provide additional data to access the interface endpoint (e.g. API URI prefix).</td>
</tr>
<tr>
<td>credentials</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>Provides credential enabling access to the interface. This property is reserved for future use in the present document.</td>
</tr>
</tbody>
</table>

### 6.2.54.3 Definition

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

```yaml
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-v1.3 [20].
Table 6.2.55.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>OauthServerInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.OauthServerInfo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.OauthServerInfo</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>BootData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:BootData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.BootData</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vim_specific_properties</td>
<td>no</td>
<td>tosca.datatypes.nfv.BootDataVimSpecificProperties</td>
<td></td>
<td>Properties used for selecting VIM specific capabilities when setting the boot data.</td>
</tr>
<tr>
<td>kvp_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.KvpData</td>
<td></td>
<td>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].</td>
</tr>
<tr>
<td>content_or_file_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.ContentOrFileData</td>
<td></td>
<td>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].</td>
</tr>
</tbody>
</table>

### 6.2.56.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>KvpData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:KvpData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.KvpData</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>A map of strings that contains a set of key-value pairs that describes the information for configuring the virtualised compute resource.</td>
</tr>
</tbody>
</table>

6.2.57.3  Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ContentOrFileData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:ContentOrFileData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.ContentOrFileData</td>
</tr>
</tbody>
</table>

#### 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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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 &quot;source_path&quot; is present and shall be absent otherwise. See note 1.</td>
</tr>
<tr>
<td>content</td>
<td>no</td>
<td>string</td>
<td></td>
<td>The string information used to customize a virtualised compute resource at boot time. See note 2.</td>
</tr>
<tr>
<td>source_path</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>destination_path</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**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:

```yaml
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.
```

*ETSI*
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>BootDataVimSpecificProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:BootDataVimSpecificProperties</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.BootDataVimSpecificProperties</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>vim_type</td>
<td>no</td>
<td>string</td>
<td>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.</td>
</tr>
<tr>
<td>properties</td>
<td>yes</td>
<td>map of string</td>
<td>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.</td>
</tr>
</tbody>
</table>

6.2.59.3 Definition

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

```yaml
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.

6.2.60 tosca.datatypes.nfv.VnfPackageChangeSelector

6.2.60.1 Description

The VnfPackageChangeSelector data type describes the source and destination VNFDs as well as source deployment flavour for a change current VNF Package. Table 6.2.60.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].
Table 6.2.60.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfPackageChangeSelector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca: VnfPackageChangeSelector</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfPackageChangeSelector</td>
</tr>
</tbody>
</table>

6.2.60.2 Properties

The properties of the VnfPackageChangeSelector data type shall comply with the provisions set out in table 6.2.60.2-1.

Table 6.2.60.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_descriptor_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the source VNFD and the source VNF package.</td>
</tr>
<tr>
<td>destination_descriptor_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the destination VNFD and the destination VNF package.</td>
</tr>
<tr>
<td>source_flavour_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the deployment flavour in the source VNF package for which this data type applies.</td>
</tr>
</tbody>
</table>

6.2.60.3 Definition

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

```javascript
tosca.datatypes.nfv.VnfPackageChangeSelector:
  derived_from: tosca.datatypes.Root
  description: data type describes the source and destination VNFDs as well as source deployment flavour for a change current VNF Package.
  properties:
    source_descriptor_id:
      type: string
      description: Identifier of the source VNFD and the source VNF package.
      required: true
    destination_descriptor_id:
      type: string
      description: Identifier of the destination VNFD and the destination VNF package.
      required: true
    source_flavour_id:
      type: string
      description: Identifier of the deployment flavour in the source VNF package for which this data type applies.
      required: true
```

6.2.60.4 Examples

See clause 6.10.15.5.

6.2.60.5 Additional Requirements

Either the source_descriptor_id or the destination_descriptor_id shall be equal to the vnfdId of the VNFD containing this version VnfPackageChangeSelector.
6.2.61  tosca.datatypes.nfv.VnfPackageChangeComponentMapping

6.2.61.1 Description

The VnfPackageChangeComponentMapping data type describes a mapping between the identifier of a components or property in the source VNFD and the identifier of the corresponding component or property in the destination VNFD. Table 6.2.61.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.61.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfPackageChangeComponentMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv: VnfPackageChangeComponentMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfPackageChangeComponentMapping</td>
</tr>
</tbody>
</table>

6.2.61.2 Properties

The properties of the VnfPackageChangeComponentMapping data type shall comply with the provisions set out in table 6.2.61.2-1.

Table 6.2.61.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_type</td>
<td>yes</td>
<td>string</td>
<td>valid_values: [vdu, cp, virtual_link, virtual_storage, instantiation_level, scaling_aspect]</td>
<td>The type of component or property. Possible values differentiate whether changes concern to some VNF component (e.g. VDU, internal VLD, etc.) or property (e.g. a Scaling Aspect, etc.).</td>
</tr>
<tr>
<td>source_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the component or property in the source VNFD.</td>
</tr>
<tr>
<td>destination_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the component or property in the destination VNFD.</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable description of the component changes.</td>
</tr>
</tbody>
</table>

6.2.61.3 Definition

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

```yaml
tosca.datatypes.nfv.VnfPackageChangeComponentMapping:
  derived_from: tosca.datatypes.Root
  description: A mapping between the identifier of a components or property in the source VNFD and the identifier of the corresponding component or property in the destination VNFD.
  properties:
    component_type:
      type: string
      description: The type of component or property. Possible values differentiate whether changes concern to some VNF component (e.g. VDU, internal VLD, etc.) or property (e.g. a Scaling Aspect, etc.).
      constraints:
        - valid_values: [vdu, cp, virtual_link, virtual_storage, instantiation_level, scaling_aspect]
      required: true
      source_id:
        type: string
        description: Identifier of the component or property in the source VNFD.
```

ETSİ
6.2.61.4 Examples
See clause 6.10.15.5.

6.2.61.5 Additional Requirements
None.

6.2.62 tosca.datatypes.nfv.VnfChangeCurrentPackageOperationConfiguration

6.2.62.1 Description
The VnfChangeCurrentPackageOperationConfiguration data type represents information that affect the invocation of the change current VNF Package operation, as specified in ETSI GS NFV-IFA 011 [1]. This data type definition is reserved for future use in the present document. Table 6.2.62.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.62.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfChangeCurrentPackageOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:VnfChangeCurrentPackageOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfChangeCurrentPackageOperationConfiguration</td>
</tr>
</tbody>
</table>

6.2.62.2 Properties
None.

6.2.62.3 Definition
The syntax of the VnfChangeCurrentPackageOperationConfiguration data type shall comply with the following definition:

tosca.datatypes.nfv.VnfChangeCurrentPackageOperationConfiguration:
  derived_from: tosca.datatypes.Root
  description: represents information that affect the invocation of the change current VNF Package operation.
  # This data type definition is reserved for future use in the present document.
  # properties:
  # derived types are expected to introduce new properties, with their type
  # derived from tosca.datatypes.nfv.VnfChangeCurrentPackageOperationConfiguration, with
  # the same name as the operation designated to the ChangeCurrentVnfPackage request
6.2.62.4 Examples

None.

6.2.62.5 Additional Requirements

None.

6.2.63 `tosca.datatypes.nfv.VnfCreateSnapshotOperationConfiguration`

6.2.63.1 Description

The `VnfCreateSnapshotOperationConfiguration` data type represents information that affect the invocation of the `CreateVnfSnapshot` operation, as specified in ETSI GS NFV-IFA 011 [1]. This data type definition is reserved for future use in the present document. Table 6.2.63.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfCreateSnapshotOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfCreateSnapshotOperationConfiguration</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfCreateSnapshotOperationConfiguration</td>
</tr>
</tbody>
</table>

6.2.63.2 Properties

None.

6.2.63.3 Definition

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

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

6.2.63.4 Examples

None.

6.2.63.5 Additional Requirements

None.

6.2.64 `tosca.datatypes.nfv.VnfRevertToSnapshotOperationConfiguration`

6.2.64.1 Description

The `VnfRevertToSnapshotOperationConfiguration` data type represents information that affect the invocation of the `RevertToVnfSnapshot` operation, as specified in ETSI GS NFV-IFA 011 [1]. This data type definition is reserved for future use in the present document. Table 6.2.64.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].
Table 6.2.64.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfRevertToSnapshotOperationConfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:VnfRevertToSnapshotOperationConfig</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfRevertToSnapshotOperationConfiguration</td>
</tr>
</tbody>
</table>

6.2.64.2 Properties

None.

6.2.64.3 Definition

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

```plaintext
tosca.datatypes.nfv.VnfRevertToSnapshotOperationConfiguration:
  derived_from: tosca.datatypes.Root
  description: represents information that affect the invocation of the
   RevertToVnfSnapshot operation
  # This data type definition is reserved for future use in the present document.
  # properties:
```

6.2.64.4 Examples

None.

6.2.64.5 Additional Requirements

None.

6.2.65 tosca.datatypes.nfv.ServicePortData

6.2.65.1 Description

The ServicePortData data type supports the specification of requirements describing port properties exposed by VirtualCp, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.65.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.65.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ServicePortData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:ServicePortData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.ServicePortData</td>
</tr>
</tbody>
</table>

6.2.65.2 Properties

The properties of the ServicePortData data type shall comply with the provisions set out in table 6.2.65.2-1.
Table 6.2.65.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>String</td>
<td></td>
<td>The name of the port exposed by the VirtualCp.</td>
</tr>
<tr>
<td>protocol</td>
<td>yes</td>
<td>String</td>
<td>Valid values: See YAML definition constraints</td>
<td>The L4 protocol for this port exposed by the VirtualCp. VALUES: • TCP • UDP • SCTP</td>
</tr>
<tr>
<td>port</td>
<td>yes</td>
<td>Integer</td>
<td></td>
<td>The L4 port number exposed by the VirtualCp.</td>
</tr>
<tr>
<td>portConfigurable</td>
<td>yes</td>
<td>Boolean</td>
<td></td>
<td>Specifies whether the port attribute value is allowed to be configurable.</td>
</tr>
</tbody>
</table>

6.2.65.3 Definition

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

```yaml
tosca.datatypes.nfv.ServicePortData:  
derived_from: tosca.datatypes.Root  
description: describes the service identifying port properties exposed by the VirtualCp  
properties:  
nome:  
  type: string  
description: The name of the port exposed by the VirtualCp.  
required: true  
protocol:  
  type: string  
description: The L4 protocol for this port exposed by the VirtualCp.  
required: true  
constraints:  
  - valid_values: [ tcp, udp, sctp ]  
port:  
  type: integer  
description: The L4 port number exposed by the VirtualCp.  
required: true  
constraints:  
  - greater_or_equal: 0  
portConfigurable:  
  type: boolean  
description: Specifies whether the port attribute value is allowed to be configurable.  
required: true
```

6.2.65.4 Examples

None.

6.2.65.5 Additional Requirements

None.
6.2.66  tosca.datatypes.nfv.AdditionalServiceData

6.2.66.1  Description

The AdditionalServiceData data type supports the specification of requirements related additional service data of the VirtualCp used to expose properties of the VirtualCp to NFV-MANO, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.66.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.66.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>AdditionalServiceData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscannfv:AdditionalServiceData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.AdditionalServiceData</td>
</tr>
</tbody>
</table>

6.2.66.2  Properties

The properties of the AdditionalServiceData data type shall comply with the provisions set out in table 6.2.66.2-1.

Table 6.2.66.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>portData</td>
<td>yes</td>
<td>List of tosca.datatypes.nfv.ServicePortData</td>
<td></td>
<td>Service port numbers exposed by the VirtualCp.</td>
</tr>
<tr>
<td>serviceData</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Service matching information exposed by the VirtualCp.</td>
</tr>
</tbody>
</table>

6.2.66.3  Definition

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

```yaml
tosca.datatypes.nfv.AdditionalServiceData:
  derived_from: tosca.datatypes.Root
  description: describes the additional service data of the VirtualCp used to expose properties of the VirtualCp to NFV-MANO.
  properties:
    portData:
      type: list
      entry_schema:
        type: tosca.datatypes.nfv.ServicePortData
        description: Service port numbers exposed by the VirtualCp.
        required: true
    serviceData:
      type: string
      description: Service matching information exposed by the VirtualCp.
      required: false
```

6.2.66.4  Examples

None.

6.2.66.5  Additional Requirements

None.
6.2.67  tosca.datatypes.nfv.VnfLcmOpCoord

6.2.67.1  Description

The VnfLcmOpCoord data type describes a set of information used for a coordination action in a VNF lifecycle management operation for a given VNF.

Table 6.2.67.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th><strong>Shorthand Name</strong></th>
<th>VnfLcmOpCoord</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type Qualified Name</strong></td>
<td>tosca.nfv:VnfLcmOpCoord</td>
</tr>
<tr>
<td><strong>Type URI</strong></td>
<td>tosca.datatypes.nfv.VnfLcmOpCoord</td>
</tr>
</tbody>
</table>

6.2.67.2  Properties

The properties of the VnfLcmOpCoord data type shall comply with the provisions set out in table 6.2.67.2-1.
<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable description of the coordination action.</td>
</tr>
<tr>
<td>endpoint_type</td>
<td>no</td>
<td>string</td>
<td>Valid values:</td>
<td>Specifies the type of the endpoint exposing the LCM operation coordination</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See YAML definition constraints</td>
<td>such as operations supporting management systems (e.g. EM) or the VNF instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values:</td>
<td>valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mgmt: coordination with other operations</td>
<td>mgmt: coordination with other operations supporting management systems (e.g. EM) or the VNF instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• vnf: coordination with the VNF instance</td>
<td>vnf: coordination with the VNF instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the VNF produces the LCM coordination</td>
<td>If the VNF produces the LCM coordination interface, this property may be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>interface, this property may be omitted or may</td>
<td>omitted or may have the value &quot;vnf&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>have the value &quot;vnf&quot;.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If this attribute is omitted, the type of</td>
<td>If this attribute is omitted, the type of endpoint that provides the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>endpoint that provides the interface is</td>
<td>interface is determined at deployment time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>determined at deployment time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the VNF does not produce the LCM</td>
<td>If the VNF does not produce the LCM coordination interface but coordination</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>coordination interface but coordination via this</td>
<td>via this interface is needed, it is expected that a management entity such as</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>interface is needed, it is expected that a</td>
<td>the EM exposes the coordination interface, and consequently, this attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>management entity such as the EM exposes the</td>
<td>shall be present and shall have the value &quot;mgmt&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>coordination interface, and consequently, this</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>attribute shall be present and shall have the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value &quot;mgmt&quot;.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the coordination action is invoked before or</td>
<td>coordination_stage property shall be omitted if the coordination action is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>after all other changes performed by the VNF</td>
<td>intended to be invoked at an intermediate stage of the LCM operation, i.e.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LCM operation.</td>
<td>neither at the start nor at the end. In this case, the time at which to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>invoke the coordination during the execution of the LCM operation is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>determined by means outside the scope of the present document such as VNF-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FM-internal logic or LCM script.</td>
</tr>
<tr>
<td>coordination_stage</td>
<td>no</td>
<td>string</td>
<td>Valid values:</td>
<td>Indicates whether the coordination action is invoked before or after all</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See YAML definition constraints</td>
<td>other changes performed by the VNF LCM operation. See note.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values:</td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• start: the coordination action is</td>
<td>• start: the coordination action is invoked after receiving the grant and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>invoked after receiving the grant and before the</td>
<td>before the LCM operation performs any other changes. end: the coordination</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LCM operation performs any other changes.</td>
<td>action is invoked after the LCM operation has performed all other changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>coordination_stage property shall be omitted if the coordination action is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>intended to be invoked at an intermediate stage of the LCM operation, i.e.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>neither at the start nor at the end. In this case, the time at which to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>invoke the coordination during the execution of the LCM operation is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>determined by means outside the scope of the present document such as VNF-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FM-internal logic or LCM script.</td>
</tr>
<tr>
<td>input_parameters</td>
<td>no</td>
<td>tosca.datatypes</td>
<td>Input parameters to be provided in the LCM</td>
<td>Input parameters to be provided in the LCM coordination request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.nfv.InputOpCo</td>
<td>coordination request</td>
<td></td>
</tr>
<tr>
<td>output_parameters</td>
<td>no</td>
<td>tosca.datatypes</td>
<td>Output parameters provided in the LCM coordination</td>
<td>Output parameters provided in the LCM coordination response</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.nfv.OutputOpC</td>
<td>response</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** The changes mentioned include changes to the VNF instance, its resources or its snapshots.
6.2.67.3 Definition

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

```yaml
tosca.datatypes.nfv.VnfLcmOpCoord:
  derived_from: tosca.datatypes.Root
  description: describes a set of information used for a coordination action in a VNF lifecycle management operation for a given VNF.
  properties:
    description:
      type: string
      description: Human readable description of the coordination action.
      required: false
    endpoint_type:
      type: string
      description: Specifies the type of the endpoint exposing the LCM operation coordination such as other operations supporting or management systems (e.g. an EM) or the VNF instance. If the VNF produces the LCM coordination interface, this property may be omitted or may have the value “vnf”. If this attribute is omitted, the type of endpoint that provides the interface is determined at deployment time. If the VNF does not produce the LCM coordination interface but coordination via this interface is needed, it is expected that a management entity such as the EM exposes the coordination interface, and consequently, this attribute shall be present and shall have the value “mgmt”.
      required: false
      constraints:
        - valid_values: [ mgmt, vnf ]
    coordination_stage:
      type: string
      description: Indicates whether the coordination action is invoked before or after all other changes performed by the VNF LCM operation. coordination_stage property shall be omitted if the coordination action is intended to be invoked at an intermediate stage of the LCM operation, i.e. neither at the start nor at the end. In this case, the time at which to invoke the coordination during the execution of the LCM operation is determined by means outside the scope of the present document such as VNFM-internal logic or LCM script.
      required: false
      constraints:
        - valid_values: [ start, end ]
  # input_parameters:
  # type: tosca.datatypes.nfv.InputOpCoordParams
  # description: Input parameters to be provided in the LCM coordination request.
  # required: false
  # output_parameters:
  # type: tosca.datatypes.nfv.OutputOpCoordParams
  # description: Output parameters provided in the LCM coordination response.
  # required: false
```

6.2.67.4 Examples

See Clause 6.10.15.

6.2.67.5 Additional Requirements

None.
6.2.68 tosca.datatypes.nfv.InputOpCoordParams

6.2.68.1 Description

The InputOpCoordParams data type is an empty base type for deriving data types for describing additional input operation coordination parameters for a given coordination action. Table 6.2.68.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>InputOpCoordParams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:InputOpCoordParams</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.InputOpCoordParams</td>
</tr>
</tbody>
</table>

6.2.68.2 Properties

None.

6.2.68.3 Definition

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

```yaml
tosca.datatypes.nfv.InputOpCoordParams:
    derived_from: tosca.datatypes.Root
    description: is an empty base type for deriving data types for describing additional input operation coordination parameters for a given coordination action
```

6.2.69 tosca.datatypes.nfv.OutputOpCoordParams

6.2.69.1 Description

The OutputOpCoordParams data type is an empty base type for deriving data types for describing additional Output operation coordination parameters for a given coordination action. Table 6.2.69.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>OutputOpCoordParams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:OutputOpCoordParams</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.OutputOpCoordParams</td>
</tr>
</tbody>
</table>

6.2.69.2 Properties

None.

6.2.69.3 Definition

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

```yaml
tosca.datatypes.nfv.OutputOpCoordParams:
    derived_from: tosca.datatypes.Root
    description: is an empty base type for deriving data types for describing additional Output operation coordination parameters for a given coordination action
```
6.2.70  tosca.datatypes.nfv.ExtendedResourceData

6.2.70.1  Description

The ExtendedResourceData data type supports the specification of requirements related to extended resources of a container, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.70.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ExtendedResourceData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:ExtendedResourceData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.ExtendedResourceData</td>
</tr>
</tbody>
</table>

6.2.70.2  Properties

The properties of the ExtendedResourceData data type shall comply with the provisions set out in table 6.2.70.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extended_resource</td>
<td>yes</td>
<td>map of string</td>
<td>length: 1</td>
<td>The hardware platform specific extended resource. A map of string that contains one single key-value pair that describes one hardware platform specific container requirement. More information regarding the usage of this property is available at: <a href="https://register.etsi.org">https://register.etsi.org</a>.</td>
</tr>
<tr>
<td>amount</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>Requested amount of the indicated extended resource.</td>
</tr>
</tbody>
</table>

6.2.70.3  Definition

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

```
tosca.datatypes.nfv.ExtendedResourceData:
  derived_from: tosca.datatypes.Root
  description: Supports the specification of requirements related to extended resources of a container.
  properties:
    extended_resource:
      type: map
      description: The hardware platform specific extended resource. A map of string that contains one single key-value pair that describes one hardware platform specific container requirement.
      required: true
    entry_schema:
      type: string
      constraints:
        - min_length: 1
        - max_length: 1
    amount:
      type: integer
      description: Requested amount of the indicated extended resource.
      required: true
```
6.2.70.4 Examples

The following example shows an OsContainer node template that contains requirements for an extended resource.

```yaml
node templates:
  container_01:
    type: tosca.nodes.nfv.Vdu.OsContainer
    properties:
      name: db_handler
      description: this container runs the process that writes and reads in the storage resource.
      requested_cpu_resources: 1
      cpu_resource_limit: 1
      requested_memory_resources: 4 MB
      memory_resource_limit: 8 MB
      extended_resource_requests:
        - extended_resource:
            dongle: |
              mandatory: "true"
            amount: 2
```

The example assumes that the capability has been registered in the NFVI Platform Capability Registry, e.g.:

<table>
<thead>
<tr>
<th>Name</th>
<th>Permitted Value</th>
<th>Version</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dongle</td>
<td>base</td>
<td>1.0</td>
<td>Generic</td>
<td>dongle accelerator</td>
</tr>
</tbody>
</table>

6.2.70.5 Additional Requirements

None.

6.2.71 tosca.datatypes.nfv.Hugepages

6.2.71.1 Description

The Hugepages data type supports the specification of requirements on a particular hugepage size in terms of total memory needs. Table 6.2.71.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hugepages</td>
<td>tosca.nfv.Hugepages</td>
<td>tosca.datatypes.nfv.Hugepages</td>
</tr>
</tbody>
</table>

6.2.71.2 Properties

The properties of the Hugepages data type shall comply with the provisions set out in table 6.2.71.2-1.
Table 6.2.71.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hugepage_size</td>
<td>yes</td>
<td>scalar-unit.size</td>
<td></td>
<td>Specifies the size of the hugepage.</td>
</tr>
<tr>
<td>requested_size</td>
<td>yes</td>
<td>scalar-unit.size</td>
<td></td>
<td>Specifies the total size required for all the hugepages of the size indicated by hugepage_size.</td>
</tr>
</tbody>
</table>

6.2.71.3 Definition

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

```
tosca.datatypes.nfv.Hugepages:
  derived_from: tosca.datatypes.Root
  description: Supports the specification of requirements on a particular hugepage size in terms of total memory needs.
  properties:
    hugepage_size:
      type: scalar-unit.size
      description: Specifies the size of the hugepage.
      required: true
    requested_size:
      type: scalar-unit.size
      description: Specifies the total size required for all the hugepages of the size indicated by hugepage_size.
      required: true
```

6.2.71.4 Examples

The following example shows an entity using a 'huge_pages_resources' property of tosca.datatypes.nfv.Hugepages data type to indicate a request of 100 MiB memory to hold hugepages of size 2 MiB, i.e. for 50 pages.

```
<some_tosca_entity>:
  properties:
    huge_pages_resources:
      hugepage_size: 2 MiB
      requested_size: 100 MiB
```

6.2.71.5 Additional Requirements

None.

6.2.72 tosca.datatypes.nfv.MaxNumberOfImpactedInstances

6.2.72.1 Description

The MaxNumberOfImpactedInstances data type specifies the maximum number of instances of a given Vdu.Compute node or VnfVirtualLink node that may be impacted simultaneously without impacting the functionality of the group of a given size. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.72.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>MaxNumberOfImpactedInstances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.MaxNumberOfImpactedInstances</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.MaxNumberOfImpactedInstances</td>
</tr>
</tbody>
</table>
6.2.72.2 Properties

The properties of the MaxNumberOfImpactedInstances data type shall comply with the provisions set out in table 6.2.72.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_size</td>
<td>no</td>
<td>integer</td>
<td></td>
<td>Determines the size of the group for which the max_number_of_impacted_instances is specified. If not present the size is not limited. See notes 1 and 2.</td>
</tr>
<tr>
<td>max_number_of_impacted_instances</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>The maximum number of instances that can be impacted simultaneously within the group of the specified size. See notes 1 and 2.</td>
</tr>
</tbody>
</table>

NOTE 1: Each group_size value specified for a group of virtual resources shall be unique, and it shall be possible to form an ascending ordered list of group sizes.

NOTE 2: The number of instances in the group for which the max_number_of_impacted_instances is specified may be equal to group_size or less. When the number of instances is less than group_size, it shall be at least 1 if this is the first group size in the ordered list of group sizes, or it shall be greater by at least 1 than the previous group size in the ordered list of group sizes.

6.2.72.3 Definition

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

```yaml
tosca.datatypes.nfv.MaxNumberOfImpactedInstances:
  derived_from: tosca.datatypes.Root
  description: Specifies the maximum number of instances of a given Vdu.Compute node or VnfVirtualLink node that may be impacted simultaneously without impacting the functionality of the group of a given size.
  properties:
    group_size:
      type: integer
      description: Determines the size of the group for which the max_number_of_impacted_instances is specified. If not present the size is not limited.
      required: false
    max_number_of_impacted_instances:
      type: integer
      description: The maximum number of instances that can be impacted simultaneously within the group of the specified size.
      required: true
```

6.2.72.4 Examples

None.

6.2.72.5 Additional Requirements

None.
6.2.73  tosca.datatypes.nfv.MinNumberOfPreservedInstances

6.2.73.1  Description

The MinNumberOfPreservedInstances data type specifies the minimum number of instances of a given Vdu.Compute node or VnfVirtualLink node which need to be preserved simultaneously. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>MinNumberOfPreservedInstances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscafv:MinNumberOfPreservedInstances</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.MinNumberOfPreservedInstances</td>
</tr>
</tbody>
</table>

Table 6.2.73.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_size</td>
<td>no</td>
<td>integer</td>
<td></td>
<td>Determines the size of the group for which the min_number_of_preserved_instances is specified. If not present the size is not limited. See notes 1 and 2.</td>
</tr>
<tr>
<td>min_number_of_preserved_instances</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>The minimum number of instances which need to be preserved simultaneously within the group of the specified size. See notes 1 and 2.</td>
</tr>
</tbody>
</table>

NOTE 1: Each group_size value specified for a group of virtual resources shall be unique, and it shall be possible to form an ascending ordered list of group sizes.

NOTE 2: The number of instances in the group for which the min_number_of_preserved_instances is specified may be equal to group_size or less. When the number of instances is less than group_size.

6.2.73.3  Definition

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

tosca.datatypes.nfv.MinNumberOfPreservedInstances:  
  derived_from: tosca.datatypes.Root  
  description: Specifies the minimum number of instances of a given Vdu.Compute node or VnfVirtualLink node which need to be preserved simultaneously.  
  properties:  
    group_size:  
      type: integer  
      description: Determines the size of the group for which the min_number_of_preserved_instances is specified. If not present the size is not limited.  
      required: false  
    min_number_of_preserved_instances:  
      type: integer  
      description: The minimum number of instances which need to be preserved simultaneously within the group of the specified size.  
      required: true
6.2.73.4  Examples

None.

6.2.73.5  Additional Requirements

None.

6.2.74  tosca.datatypes.nfv.NfviMaintenanceInfo

6.2.74.1  Description

The NfviMaintenanceInfo data type provides information related to the constraints and rules applicable to virtualised resources and their groups impacted due to NFVI maintenance operations, as defined in ETSI GS NFV-IFA 011 [1]. Table 6.2.74.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.2.74.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NfviMaintenanceInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscaf:nfvi:NfviMaintenanceInfo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NfviMaintenanceInfo</td>
</tr>
</tbody>
</table>

6.2.74.2  Properties

The properties of the NfviMaintenanceInfo data type shall comply with the provisions set out in table 6.2.74.2-1.

Table 6.2.74.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>impact_notification_lead_time</td>
<td>yes</td>
<td>scalar-unit.time</td>
<td></td>
<td>Specifies the minimum notification lead time requested for upcoming impact of the virtualised resource or their group (i.e. between the notification and the action causing the impact).</td>
</tr>
<tr>
<td>is_impact_mitigation_requested</td>
<td>yes</td>
<td>boolean</td>
<td>default: false</td>
<td>Indicates whether it is requested that at the time of the notification of an upcoming change that is expected to have an impact on the VNF, virtualised resource(s) of the same characteristics as the impacted ones is/are provided to compensate for the impact (TRUE) or not (FALSE).</td>
</tr>
<tr>
<td>supported_migration_type</td>
<td>no</td>
<td>list of string</td>
<td>valid values: no_migration, offline_migration, live_migration</td>
<td>Specifies the allowed migration types in the order of preference in case of an impact starting with the most preferred type. It is applicable to the Vdu.Compute node and to the VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes. See note 1.</td>
</tr>
<tr>
<td>max_undetectable INTERRUPTION_TIME</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>Specifies the maximum interruption time that can go undetected at the VNF level and therefore which will not trigger VNF-internal recovery during live migration. It is applicable to the Vdu.Compute node and to the VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes. See note 1.</td>
</tr>
<tr>
<td>min_recovery_time_between_impacts</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>Specifies the time required by the group to recover from an impact, thus, the minimum time requested between consecutive impacts of the group. See note 2.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>max_number_of_impacted_instances</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.MaxNumberOfImpactedInstances</td>
<td></td>
<td>Specifies for different group sizes the maximum number of instances that can be impacted simultaneously within the group of virtualised resources without losing functionality. See notes 2 and 3.</td>
</tr>
<tr>
<td>min_number_of_preserved_instances</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.MinNumberOfPreservedInstances</td>
<td></td>
<td>Specifies for different group sizes the minimum number of instances which need to be preserved simultaneously within the group of virtualised resources. See notes 2 and 3.</td>
</tr>
</tbody>
</table>

**NOTE 1:** When the max_undetectable_interruption_time is specified it constrains the live migration. If it cannot be guaranteed on an NFVI that the interruption caused by the live migration will be less than the indicated maximum undetectable interruption time, then life migration should be downgraded according to the order of preference.

**NOTE 2:** Impacts to instances of the group happening within the min_recovery_time_between_impacts are considered simultaneous impacts.

**NOTE 3:** Either “max_number_of_impacted_instances” or “min_number_of_preserved_instances” may be provided, but not both.

### 6.2.74.3 Definition

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

```yaml
tosca.datatypes.nfv.NfviMaintenanceInfo:
  derived_from: tosca.datatypes.Root
  description: Provides information related to the constraints and rules applicable to virtualised resources and their groups impacted due to NFVI maintenance operations
  properties:
    impact_notification_lead_time:
      type: scalar-unit.time
      description: Specifies the minimum notification lead time requested for upcoming impact of the virtualised resource or their group (i.e. between the notification and the action causing the impact).
      required: true
    is_impact_mitigation_requested:
      type: boolean
      description: Indicates whether it is requested that at the time of the notification of an upcoming change that is expected to have an impact on the VNF, virtualised resource(s) of the same characteristics as the impacted ones is/are provided to compensate for the impact (TRUE) or not (FALSE).
      required: true
      default: false
    supported_migration_type:
      type: list
      description: Specifies the allowed migration types in the order of preference in case of an impact starting with the most preferred type. It is applicable to the Vdu.Compute node and to the VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes.
      required: false
      entry_schema:
        type: string
      constraints:
        - valid_values: [ no_migration, offline_migration, live_migration ]
    max_undetectable_interruption_time:
      type: scalar-unit.time
      description: Specifies the maximum interruption time that can go undetected at the VNF level and therefore which will not trigger VNF-internal recovery during live migration. It is applicable to the Vdu.Compute node and to the VirtualBlockStorage, VirtualObjectStorage and VirtualFileStorage nodes.
      required: false
    min_recovery_time_between_impacts:
```

---

**ETSI**
6.2.74.4 Examples

None.

6.2.74.5 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>SwImage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:SwImage</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.artifacts.nfv.SwImage</td>
</tr>
</tbody>
</table>

6.3.1.2 Properties

The properties of the SwImage artifacts type shall comply with the provisions set out in table 6.3.1.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Name of this software image.</td>
</tr>
<tr>
<td>version</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Version of this software image.</td>
</tr>
<tr>
<td>provider</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provider of this software image.</td>
</tr>
</tbody>
</table>
### 6.3.1.3 Definition

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

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: false
    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: false
    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
6.3.2 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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Mistral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.Mistral</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.artifacts.Implementation.nfv.Mistral</td>
</tr>
<tr>
<td>derived_from</td>
<td>tosca.artifacts.Implementation</td>
</tr>
</tbody>
</table>

6.3.2.2 Definition

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

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

6.3.3 tosca.artifacts.nfv.HelmChart

6.3.3.1 Description

The HelmChart artifact is a file containing a HelmTM chart [23].

Table 6.3.3.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Whether the Helm chart contains custom resource definitions is out of scope of the present document.

Table 6.3.3.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>HelmChart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.HelmChart</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.artifacts.nfv.HelmChart</td>
</tr>
</tbody>
</table>

6.3.3.2 Properties

None.

6.3.3.3 Description

```yaml
tosca.artifacts.nfv.HelmChart:
  derived_from: tosca.artifacts.File
  description: describes the Helm chart artifact.
  file_ext: [ tar, tar.gz, tgz ]
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualBindable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VirtualBindable</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.VirtualBindable</td>
</tr>
</tbody>
</table>

6.4.1.2 Properties
None.

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualCompute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VirtualCompute</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.VirtualCompute</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical_node</td>
<td>no</td>
<td>map of tosca.datatypes.nfv.LogicalNodeData</td>
<td></td>
<td>The Logical Node requirements.</td>
</tr>
<tr>
<td>request_additional_capabilities</td>
<td>no</td>
<td>map of tosca.datatypes.nfv.RequestedAdditionalCapability</td>
<td></td>
<td>Describes additional capability for a particular VDU.</td>
</tr>
<tr>
<td>compute_requirements</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>Describes compute requirements.</td>
</tr>
<tr>
<td>virtual_memory</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VirtualMemory</td>
<td></td>
<td>Describes virtual memory of the virtualised compute.</td>
</tr>
<tr>
<td>virtual_cpu</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VirtualCpu</td>
<td></td>
<td>Describes virtual CPU(s) of the virtualised compute.</td>
</tr>
<tr>
<td>virtual_local_storage</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.VirtualBlockStorageData</td>
<td></td>
<td>A list of virtual system disks created and destroyed as part of the VM lifecycle.</td>
</tr>
</tbody>
</table>

6.4.3.3 Definition

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

```yaml
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
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualStorage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VirtualStorage</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
</tr>
</tbody>
</table>

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

6.4.5  tosca.capabilities.nfv.TrunkBindable

6.4.5.1  Description
The TrunkBindable capability indicates that the VduCp node that includes it can be pointed by a
tosca.relationships.nfv.TrunkBindsTo relationship type which is used to model the trunkPortTopology illustrated in ETSI GS NFV-IFA 011 [1]. Table 6.4.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>TrunkBindable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:TrunkBindable</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.TrunkBindable</td>
</tr>
</tbody>
</table>

6.4.5.2  Properties
None.

6.4.5.3  Definition
The syntax of the TrunkBindable capability type shall comply with the following definition:

tosca.capabilities.nfv.TrunkBindable:
derived_from: tosca.capabilities.Node
description: Indicates that the node that includes it can be pointed by a tosca.relationships.nfv.TrunkBindsTo relationship type which is used to model the trunkPortTopology.

6.4.6 tosca.capabilities.nfv.ContainerDeployable

6.4.6.1 Description

A node type that includes the ContainerDeployable capability indicates that it can be pointed by tosca.relationships.nfv.DeploysTo relationship type, which is used to model the grouping of Vdu.OsContainers into a Vdu.OsContainerDeployableUnit. Table 6.4.6.1-1 specifies the declared names for this capability type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ContainerDeployable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscafvn:ContainerDeployable</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.ContainerDeployable</td>
</tr>
</tbody>
</table>

6.4.6.2 Properties

None.

6.4.6.3 Definition

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

tosca.capabilities.nfv.ContainerDeployable:
  derived_from: tosca.capabilities.Node
  description: A node type that includes the ContainerDeployable capability indicates that it can be pointed by tosca.relationships.nfv.DeploysTo relationship type

6.4.7 tosca.capabilities.nfv.AssociableVdu

6.4.7.1 Description

The AssociableVdu capability indicates that the node that includes it can be pointed by a tosca.relationships.nfv.MciopAssociates relationship type which is used to model the associatedVdu property of the MciopProfile information element defined in ETSI GS NFV-IFA 011 [1]. Table 6.4.7.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>AssociableVdu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscafvn:AssociableVdu</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.AssociableVdu</td>
</tr>
</tbody>
</table>

6.4.7.2 Properties

None.
6.4.7.3 Definition

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

```yaml
tosca.capabilities.nfv.AssociableVdu:
  derived_from: tosca.capabilities.Node
  description: Indicates that the node that includes it can be pointed by a tosca.relationships.nfv.MciopAssociates relationship type which is used to model the associatedVdu property of the MciopProfile information element defined in ETSI GS NFV-IFA 011 [1].
```

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 or Vdu.OsContainerDeployableUnit 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualBindsTo</td>
<td>tosca.nfv:VirtualBindsTo</td>
<td>tosca.relationships.nfv.VirtualBindsTo</td>
</tr>
</tbody>
</table>

6.6.1.2 Properties

None.

6.6.1.3 Definition

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

```yaml
tosca.relationships.nfv.VirtualBindsTo:
  derived_from: tosca.relationships.DependsOn
  description: Represents an association relationship between Vdu.Compute or Vdu.OsContainerDeployableUnit 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 or Vdu.OsContainerDeployableUnit 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttachesTo</td>
<td>tosca.relationships.nfv.AttachesTo</td>
<td>tosca.relationships.nfv.AttachesTo</td>
</tr>
</tbody>
</table>

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

6.6.3.2  Properties

None.

6.6.3.3  Definition

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

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

6.6.4  tosca.relationships.nfv.TrunkBindsTo

6.6.4.1  Description

This relationship type represents an association between a VduCp node used as a trunk port and other VduSubCp nodes used as subports of the same trunk. Table 6.6.4.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrunkBindsTo</td>
<td>tosca.relationships.nfv.TrunkBindsTo</td>
<td>tosca.relationships.nfv.TrunkBindsTo</td>
</tr>
</tbody>
</table>

Table 6.6.4.1-1: Type name, shorthand, and URI

6.6.4.2  Properties

None.

6.6.4.3  Definition

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

```yaml
tosca.relationships.nfv.TrunkBindsTo:
  derived_from: tosca.relationships.DependsOn
  description: Represents the association relationship between a VduCp node used as a trunk port and other VduSubCp nodes used as subports of the same trunk.
  valid_target_types: [ tosca.capabilities.nfv.TrunkBindable ]
```
6.6.5 tosca.relationships.nfv.DeploysTo

6.6.5.1 Description

This relationship type represents an association between Vdu.OsContainerDeployableUnit and Vdu.OsContainer node types. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.6.5.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>DeploysTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv: DeploysTo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.relationships.nfv.DeploysTo</td>
</tr>
</tbody>
</table>

6.6.5.2 Properties

None.

6.6.5.3 Definition

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

```yaml
tosca.relationships.nfv.DeploysTo:
  derived_from: tosca.relationships.DependsOn
  description: Represents an association relationship between Vdu.OsContainerDeployableUnit and Vdu.OsContainer node types
  valid_target_types: [tosca.capabilities.nfv.ContainerDeployable]
```

6.6.6 tosca.relationships.nfv.MciopAssociates

6.6.6.1 Description

This relationship type represents an association between Mciop and Vdu.OsContainerDeployableUnit node types. Table 6.6.6.1-1 specifies the declared names for this relationship type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.6.6.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>MciopAssociates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.MciopAssociates</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.relationships.nfv.MciopAssociates</td>
</tr>
</tbody>
</table>

6.6.6.2 Properties

None.

6.6.6.3 Definition

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

```yaml
tosca.relationships.nfv.MciopAssociates:
  derived_from: tosca.relationships.DependsOn
  description: Represents an association relationship between Mciop and Vdu.OsContainerDeployableUnit node types
  valid_target_types: [tosca.capabilities.nfv.AssociableVdu]
```
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
- Create VNF Snapshot
- Revert to VNF Snapshot
- Change VNF current package

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.

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

- Change VNF current package

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

- `<base_notification_name>_start` for a preamble
- `<base_notification_name>_end` for a postamble

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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vnflcm</td>
<td>tosca.interfaces.nfv.Vnflcm</td>
<td>tosca.interfaces.nfv.Vnflcm</td>
</tr>
</tbody>
</table>
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.
  operations:
    instantiate:
      description: Invoked upon receipt of an Instantiate VNF request
      # inputs:
      # additional_parameters:
        # type: tosca.datatypes.nfv.VnfOperationAdditionalParameters
        # required: false
    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
    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
    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
    change_external_connectivity_start:
      description: Invoked before change_external_connectivity
    change_external_connectivity_end:
      description: Invoked after change_external_connectivity
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
inputs:
  cause:
    type: string
    description: Indicates the reason why a healing procedure is required.
    required: false
  vnfc_instance_ids:
    type: list
    entry_schema:
      type: string
      description: List of VNFC instances requiring a healing action.
      required: false
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
inputs:
  type:
    type: string
    description: Indicates the type of the scale operation requested.
    required: false
    constraints:
      - valid_values: [ scale_out, scale_in ]
  aspect:
    type: string
    description: Identifier of the scaling aspect.
    required: false
  number_of_steps:
    type: integer
    description: Number of scaling steps to be executed.
required: true
c
constraints:
  - greater_than: 0
default: 1
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
inputs:
  instantiation_level:
    type: string
description: Identifier of the target instantiation level of the current
deployment flavour to which the VNF is requested to be scaled. Either
instantiation_level or scale_info shall be provided.
  required: false
  scale_info:
    type: map # key: aspectId
description: For each scaling aspect of the current deployment flavour,
indicates the target scale level to which the VNF is to be scaled. Either
instantiation_level or scale_info shall be provided.
  required: false
  entry_schema:
    type: tosca.datatypes.nfv.ScaleInfo
scale_to_level_start:
  description: Invoked before scale_to_level
scale_to_level_end:
  description: Invoked after scale_to_level
create_snapshot:
  description: Invoked upon receipt of a Create VNF snapshot 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
create_snapshot_start:
  description: Invoked before create_snapshot
create_snapshot_end:
  description: Invoked after create_snapshot
revert_to_snapshot:
  description: Invoked upon receipt of a Revert to VNF snapshot 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
revert_to_snapshot_start:
  description: Invoked before revert_to_snapshot
revert_to_snapshot_end:
  description: Invoked after revert_to_snapshot
change_current_package:
  description: Invoked by tosca.policies.nfv.VnfPackageChange
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-v1.3 [20] 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.

Starting with version 3.3.1 of the present document, the Vnflcm interface type definition grammar was changed to support notifications and operations. For backward compatibility, TOSCA-Simple-Profile-YAML-v1.3 [20], clause 3.7.5.5 specifies the provisions for handling the previous grammar. Support of the Release 2 Vnflcm interface type definition grammar can be removed in subsequent versions of the present document.
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-v1.3 [20]. 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.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
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:
          operations:
            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).
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.
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).

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3

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:
      operations:
        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.VnfIndicator 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>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfIndicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:VnfIndicator</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.interfaces.nfv.VnfIndicator</td>
</tr>
</tbody>
</table>

6.7.2.2  Definition

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

```yaml
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.

6.7.3  tosca.interfaces.nfv.ChangeCurrentVnfPackage

6.7.3.1  Description

The tosca.interfaces.nfv.ChangeCurrentVnfPackage is an empty base interface type for deriving VNF specific interface types that include VNF Change Current VNF Package specific operation.

Table 6.7.3.1-1 specifies the declared names for this interface type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ChangeCurrentVnfPackage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:ChangeCurrentVnfPackage</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.interfaces.nfv.ChangeCurrentVnfPackage</td>
</tr>
</tbody>
</table>

6.7.3.2  Definition

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

```yaml
tosca.interfaces.nfv.ChangeCurrentVnfPackage:
  derived_from: tosca.interfaces.Root
  description: This interface is an empty base interface type for deriving VNF specific interface types that include VNF Change Current VNF Package specific operation.

  # operations:
    # operation_name: name of a VNF-specific operation serving the Change current VNF Package request.
    # description: Invoked by tosca.policies.nfv.VnfPackageChange
```

---

ETSİ
# 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

6.7.3.3 Examples
See clause 6.10.15.5.

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Table 6.8.1.1-1: Type name, shorthand, and URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shorthand Name</td>
</tr>
<tr>
<td>Type Qualified Name</td>
</tr>
<tr>
<td>Type URI</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table 6.8.1.2-1: Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>descriptor_id</td>
</tr>
<tr>
<td>descriptor_version</td>
</tr>
<tr>
<td>provider</td>
</tr>
<tr>
<td>product_name</td>
</tr>
<tr>
<td>software_version</td>
</tr>
<tr>
<td>product_info_name</td>
</tr>
<tr>
<td>product_info_description</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>vnfm_info</td>
</tr>
<tr>
<td>localization_languages</td>
</tr>
<tr>
<td>default_localization_language</td>
</tr>
<tr>
<td>configurable_properties</td>
</tr>
<tr>
<td>modifiable_attributes</td>
</tr>
<tr>
<td>lcm_operations_configuration</td>
</tr>
<tr>
<td>monitoring_parameters</td>
</tr>
<tr>
<td>flavour_id</td>
</tr>
<tr>
<td>flavour_description</td>
</tr>
<tr>
<td>vnf_profile</td>
</tr>
</tbody>
</table>
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\[0-9\]\.?\[0-9\]\.?\[0-9\]\.?\[0-9\])|(\[0-9\]+:\[a-zA-Z0-9.-\]+)$

### 6.8.1.3 Attributes

The attribute of the VNF node type shall comply with the provisions set out in table 6.8.1.3-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale_status</td>
<td>no</td>
<td>map of tosca.datatypes.nfv. ScaleInfo</td>
<td></td>
<td>Scale status of the VNF, one entry per aspect. Represents for every scaling aspect how &quot;big&quot; the VNF has been scaled w.r.t. that aspect.</td>
</tr>
</tbody>
</table>

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.

**NOTE 1:** 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.

If the scale_status attribute is used in a VNF indicator policy, e.g. an auto-scale policy, the VNF specific node type definitions may include additional attribute definitions of type integer, one for each scaling aspect. See example in clause A.15.

**NOTE 2:** As the scale_status attribute is complex, the scale_level property of the individual scaling aspects can be retrieved by passing a path to the get_attribute function: [ get_attribute: [ SELF, scale_status, [scaling_aspect], scale_level ] ]. If the value of the scale_level property is needed in a constraint (tosca.policies.nfv.VnfIndicator), then the value can be retrieved in an indirect way by accessing the aforementioned additional attributes. This is due to the limitation mentioned in the previous note.

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-YAML-v1.3 [20] 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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link</td>
</tr>
</tbody>
</table>

### 6.8.1.5 Capabilities

None.

### 6.8.1.6 Definition

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

```yaml
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
contstraints:
  - pattern: (^[etsivnfm:v[0-9]+[0-9]+[0-9]+[0-9]+[0-9]+[0-9]+[0-9]+[0-9]+]$)+

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: map # key: id
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
attributes:
scale_status:
type: map # key: aspectId
description: Scale status of the VNF, one entry per aspect. Represents for every scaling aspect how "big" the VNF has been scaled w.r.t. that aspect.
entry_schema:
type: tosca.datatypes.nfv.ScaleInfo
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:

a) The node type shall be derived from: tosca.nodes.nfv.VNF.

b) 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:
   a. descriptor_id
   b. descriptor_version
   c. provider
   d. product_name
   e. software_version
   f. vnfm_info
   g. flavour_id

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

c) An empty string shall be indicated as the default value of the flavour_description property, without providing constraints.

d) The capabilities, requirements, interfaces of tosca.nodes.nfv.VNF shall be preserved.

e) Depending on the number of external connection points of the VNF that need to connect to NS virtual links, additional requirements for VirtualLinkable capability shall be defined 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.
f) The rule for naming this node type in the service template should be:
   - provider.product_name.software_version.descriptor_version, by concatenating the values of the corresponding properties of the created VNF node type.

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

g) 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 and, optionally, tosca.policies.nfv.VnfIndicator policies that may invoke auto-scale or auto-heal operations. 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].

h) If "additionalParams" are expected in the Change current VNF Package request on the API (NFV-SOL 003 or NFV-SOL 002), then they shall be defined as "additional_parameters" inputs of the change_current_package operation on the Vnflcm interface (in case the same LCM script with the same set of "additionalParams" is suitable for all change paths) or the VNF-specific operations on the ChangeCurrentVnfPackage interface (in case different change paths require different LCM scripts potentially with different sets of "additionalParams").

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.VnfInfoModifiableAttributes.

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:

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
node_types:
  MyCompany.SunshineDB.1_0.1_0:
```
Example usage of lcm_operations_configuration:

Top level service template:

tosca_definitions_version: tosca_simple_yaml_1_3

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:

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
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:

```yaml
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:
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfExtCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.VnfExtCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.VnfExtCp</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_network_interface_requirements</td>
<td>no</td>
<td>list of tosca.datatype.s.nfv.VirtualNetworkInterfaceRequirements</td>
<td>The actual virtual NIC requirements that is been assigned when instantiating the connection point.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>external_virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability</td>
</tr>
<tr>
<td>internal_virtual_link</td>
<td>yes</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability</td>
</tr>
</tbody>
</table>

6.8.2.5 Capabilities

None.

6.8.2.6 Definition

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

```python
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) or a VipCp or a VirtualCp or a VduSubCp, the VduCp or VipCp or VirtualCp or VduSubCp 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 include 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.

When a VNF external connection point re-exposes a VIP connection point, the service template does not include 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 VipCp node. This example is illustrated in clause A.13.

When a VNF external connection point re-exposes a Subport connection point in the trunk mode, the service template does not include 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 VduSubCp node.

When a VNF external connection point re-exposes a Virtual connection point, the service template does not include 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 VirtualCp node. This example is illustrated in clause A.18.

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 (when realized as a VM) 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Vdu.Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:ns.nfv.Vdu.Compute</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.Vdu.Compute</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td></td>
<td>Human readable name of the Vdu.</td>
</tr>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td></td>
<td>Human readable description of the Vdu.</td>
</tr>
<tr>
<td>boot_order</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Status</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>-------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>nfvi_constraints</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>monitoring_parameters</td>
<td>no</td>
<td>map of tosca.data types.nfv. VnfcMonitoringParameter</td>
<td></td>
<td></td>
<td>Describes monitoring parameters applicable to a VNFC based on this VDU.</td>
</tr>
<tr>
<td>configurable_properties</td>
<td>no</td>
<td>tosca.data types.nfv. VnfcConfigurableProperties</td>
<td></td>
<td></td>
<td>Describes the configurable properties of all VNFC instances based on this VDU.</td>
</tr>
<tr>
<td>boot_data</td>
<td>no</td>
<td>tosca.data types.nfv. BootData</td>
<td></td>
<td></td>
<td>Contains the information used to customize a virtualised compute resource at boot time. See note. The bootData may contain variable parts that are replaced by deployment specific values before being sent to the VIM. For &quot;volatile&quot; parameters, i.e. those that exist only during the lifetime of an LCM operation occurrence, the parameters of each variable part shall be declared in a type derived from tosca.datatypes.nfv.VnfOperationAdditio nalParameters. For &quot;persistent&quot; parameters, i.e. those that exist during the lifetime of the VNF instance beyond the lifetime of a single LCM operation occurrence, the parameters shall be declared in a type derived from tosca.datatypes.nfv.VnfInfoModifiableAttributesExtensions or tosca.datatypes.nfv.VnfConfigurableProperties.</td>
</tr>
<tr>
<td>vdu_profile</td>
<td>yes</td>
<td>tosca.data types.nfv. VduProfile</td>
<td></td>
<td></td>
<td>Defines additional instantiation data for the VDU.Compute node.</td>
</tr>
</tbody>
</table>

NOTE: The boot_data structure passed to a VNFC instance cannot be changed after the boot time of the VNFC instance.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_storage</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
<td></td>
<td>Describes storage requirements for a virtual_storage instance attached to the virtualisation container created from virtual_compute defined for this vdu</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_compute</td>
<td>tosca.capabilities.nfv.VirtualCompute</td>
<td></td>
<td>Describes virtual compute resources capabilities.</td>
</tr>
<tr>
<td>virtual_binding</td>
<td>tosca.capabilities.nfv.VirtualBindable</td>
<td></td>
<td>Defines ability of VirtualBindable.</td>
</tr>
</tbody>
</table>

6.8.3.6 Definition

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

```yaml
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: map # key: id
      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.VnfConfigurableProperties
    #required: false
    # derived types are expected to introduce
    # configurable_properties with its type derived from
    # tosca.datatypes.nfv.VnfConfigurableProperties
    vdu_profile:
      type: tosca.datatypes.nfv.VduProfile
      description: Defines additional instantiation data for the VDU.Compute node
```
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.VnfcAdditionalConfigurableProperties 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 by using modifiable_attributes.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
...
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:
    extensions:
      type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions

substitution_mappings:
```
This example illustrates boot data containing kvp_data by using configurable_properties.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
  ...
node_types:
  mycompany.nodes.nfv.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      
  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
    required: false

  mycompany.datatypes.nfv.VnfAdditionalConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties
    properties:
      host_name:
        type: string
        required: false

topology_template:
  inputs:
    configurable_properties:
      type: mycompany.datatypes.nfv.VnfConfigurableProperties

substitution_mappings:
  node_type: mycompany.nodes.nfv.SunshineDB.1_0.1_0
  ...
```
This example illustrates fetching the boot data value by using `content_or_file_data`.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
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:
node_templates:
  vnf:
    type: mycompany.nodes.nfv.SunshineDB.1_0.1_0
    properties:
      ..
      configurable_properties: { get_input: configurable_properties }
  dbBackend:
    type: tosca.nodes.nfv.Vdu.Compute
    properties:
      ..
      boot_data:
        kvp_data:
          data:
            ip_address_1: { get_property: [vnf, configurable_properties, additional_configurable_properties, host_name ] }

  ..
```
extensions:
  type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions

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:
        extensions: { get_input: extensions }

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.

tosca_definitions_version: tosca_simple_yam1_1_3
..
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 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_3
..
topology_template:
  ...
  node_templates:
    dbBackend:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        ...
        ...
        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 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_3
..
topology_template:
  ...
  node_templates:
    dbBackend:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        ...
        ...
        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
            ...
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualBlockStorage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VirtualBlockStorage</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.Vdu.VirtualBlockStorage</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_block_stor</td>
<td>yes</td>
<td>tosca.datatypes.nfv.V</td>
<td></td>
<td></td>
<td>Describes the block storage characteristics.</td>
</tr>
<tr>
<td>age_data</td>
<td></td>
<td>age_blockStorage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>per_vnf_instance</td>
<td>yes</td>
<td>boolean</td>
<td>default: true</td>
<td></td>
<td>Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.</td>
</tr>
<tr>
<td>nfvi_maintenance_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.Nf</td>
<td></td>
<td></td>
<td>Provides information on the rules to be observed when an instance based on this VirtualBlockStorage is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).</td>
</tr>
<tr>
<td>vi Maintena</td>
<td></td>
<td>nceInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_storage</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
<td></td>
<td>Defines the capabilities of virtual_storage.</td>
</tr>
</tbody>
</table>

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

```python
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
      per_vnfc_instance:
         type: boolean
         description: Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.
         required: true
         default: true
      nfvi_maintenance_info:
         type: tosca.datatypes.nfv.NfviMaintenanceInfo
         description: Provides information on the rules to be observed when an instance based on this VirtualBlockStorage is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).
         required: false
   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 when attached to the node with type tosca.nodes.nfv.Vdu.Compute, otherwise, such artifact definition shall not be present. 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualObjectStorage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VirtualObjectStorage</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.Vdu.VirtualObjectStorage</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_object_storage_data</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VirtualObjectStorageData</td>
<td></td>
<td>Describes the object storage characteristics.</td>
</tr>
<tr>
<td>per_vnfc_instance</td>
<td>yes</td>
<td>boolean</td>
<td>default: true</td>
<td>Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.</td>
</tr>
<tr>
<td>nfvi_maintenance_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.NfviMaintenanceInfo</td>
<td></td>
<td>Provides information on the rules to be observed when an instance based on this VirtualObjectStorage is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_storage</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
<td></td>
<td>Defines the capabilities of virtual_storage.</td>
</tr>
</tbody>
</table>

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
per_vnfc_instance:
  type: boolean
description: Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.
required: true
default: true
nfvi_maintenance_info:
  type: tosca.datatypes.nfv.NfviMaintenanceInfo
description: Provides information on the rules to be observed when an instance based on this VirtualObjectStorage is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).
required: false
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_file_storage_ data</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VirtualFileStorageData</td>
<td></td>
<td>Describes the file storage characteristics.</td>
</tr>
<tr>
<td>per_vnfc_instance</td>
<td>yes</td>
<td>boolean</td>
<td>default: true</td>
<td>Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.</td>
</tr>
</tbody>
</table>
### 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_link</td>
<td>yes</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link</td>
</tr>
</tbody>
</table>

### 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_storage</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
<td></td>
<td>Defines the capabilities of virtual_storage.</td>
</tr>
</tbody>
</table>
6.8.6.6 Definition

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

```yaml
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
    per_vnfc_instance:
      type: boolean
      description: Indicates whether the virtual storage descriptor shall be instantiated per VNFC instance.
      required: true
      default: true
    nfvi_maintenance_info:
      type: tosca.datatypes.nfv.NfviMaintenanceInfo
      description: Provides information on the rules to be observed when an instance based on this VirtualFileStorage is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).
      required: false
  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-v1.3 [20].
Table 6.8.8.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfV:VduCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfV:VduCp</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitrate_requirement</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Bitrate requirement in bit per second on this connection point. See note 2.</td>
</tr>
<tr>
<td>virtual_network_interface_requirements</td>
<td>no</td>
<td>list</td>
<td></td>
<td>Specifies requirements on a virtual network interface realizing the CPs instantiated from this CPD. See note 2.</td>
</tr>
<tr>
<td>order</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>The order of the NIC on the compute instance (e.g. eth2). See note 1.</td>
</tr>
<tr>
<td>vnic_type</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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. Additional values of the attribute for VDUs realized by one or set of OS containers: bridge, ipvlan, loopback, macvlan, ptp, vlan, host-device</td>
</tr>
</tbody>
</table>

NOTE 1: When binding more than one port to a single compute (aka multi vNICs) and ordering is desired, it is mandatory that all ports will be set with an order value. The order values shall represent a positive, arithmetic progression that starts with 0 (i.e. 0, 1, 2,..., n).

NOTE 2: For VDUs realized by one or a set of OS containers, the ability to configure virtualised resources based on this property might not be supported by all container technologies.

NOTE 3: For CPs of VDUs realized by one or set of OS containers and used by the OS containers to connect to the primary container cluster external network, the ability to configure virtualised resources based on cpRole and trunkMode properties might not be supported by all container technologies.

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_binding</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualBindable</td>
<td>Describe the requirement for binding with VDU</td>
<td></td>
</tr>
<tr>
<td>virtual_link</td>
<td>yes</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td>Describes the requirements for linking to virtual link</td>
<td></td>
</tr>
</tbody>
</table>

6.8.8.5 Capabilities

The capabilities of the VduCp node type shall comply with the provisions set out in table 6.8.8.5-1. This capability is available only the trunk_mode property value of this VduCp is "true" and there is at least one VduSubCp defined as subport of the same trunk.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunk_binding</td>
<td>tosca.capabilities.nfv.TrunkBindable</td>
<td></td>
<td>Defines ability of TrunkBindable.</td>
</tr>
</tbody>
</table>

6.8.8.6 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, bridge, ipvlan, loopback, macvlan, ptp, vlan, host-device ]
  capabilities:
  trunk_binding: # This capability is available only the trunk_mode property value of this VduCp is true and there is at least one VduSubCp defined as subport of the same trunk.
  type: tosca.capabilities.nfv.TrunkBindable
  occurrences: [ 0, UNBOUNDED ]
requirements:
- virtual_link:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo
  occurrences: [ 1, 1 ]
- virtual_binding:
  capability: tosca.capabilities.nfv.VirtualBindable
  relationship: tosca.relationships.nfv.VirtualBindsTo
  occurrences: [ 0, 1 ]

6.8.8.7 Additional Requirements

The occurrence 0 of the virtual_binding requirement is applicable for node templates of tosca.nodes.nfv.VduSubCp node type derived from tosca.nodes.nfv.VduCp. For node templates of tosca.nodes.nfv.VduCp node type occurrence 1 applies.

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfVirtualLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VnfVirtualLink</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.VnfVirtualLink</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectivity_type</td>
<td>yes</td>
<td>ConnectivityType</td>
<td></td>
<td>Specifies the protocol exposed by the VL and the flow pattern supported by the VL.</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provides human-readable information on the purpose of the VL (e.g. control plane traffic).</td>
</tr>
<tr>
<td>test_access</td>
<td>no</td>
<td>list of string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Test access facilities available on the VL.</td>
</tr>
<tr>
<td>vl_profile</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VIPProfile</td>
<td></td>
<td>Defines additional data for the VL: maximum and minimum bit rate requirements and QoS.</td>
</tr>
<tr>
<td>monitoring_parameters</td>
<td>no</td>
<td>map of tosca.datatypes.nfv.VirtualLinkMonitoring Parameter</td>
<td></td>
<td>Describe monitoring parameters applicable to a VL instantiated from this node type. See note.</td>
</tr>
<tr>
<td>nfvi_maintenance_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.NfviMaintenanceInfo</td>
<td></td>
<td>Provides information on the rules to be observed when an instance based on this VnfVirtualLink is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).</td>
</tr>
</tbody>
</table>

NOTE: This property shall not be present in a VNFD service template when all the virtualisation containers of the VNF are realized as OsContainers.

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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_linkable</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Defines ability of VirtualLinkable.</td>
</tr>
</tbody>
</table>

6.8.9.5 Definition

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

```yaml
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
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VipCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:VipCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.VipCp</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dedicated_ip_address</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>Indicates whether the VIP address shall be different from the addresses allocated to all associated VduCp instances or shall be the same as one of them. If set to true, the VIP address shall be different from the addresses allocated to all of the VduCp instances associated to it. If set to false, the VIP address shall be the same as one of the VduCp instances associated to it.</td>
</tr>
<tr>
<td>vip_function</td>
<td>yes</td>
<td>string</td>
<td>valid values:</td>
<td>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, 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.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>yes</td>
<td>tosca.capabilities. Node</td>
<td></td>
<td>Describes the requirement for connecting to VDU CP instances that share the virtual IP address.</td>
</tr>
<tr>
<td>virtual_link</td>
<td>yes</td>
<td>tosca.capabilities. nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link.</td>
</tr>
</tbody>
</table>

### 6.8.10.5 Definition

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

```yaml
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:
    dedicated_ip_address:
      type: boolean
      description: Indicates whether the VIP address shall be different from the addresses allocated to all associated VduCp instances or shall be the same as one of them.
      required: true
      default: true
    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, 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." |
```
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
  node: tosca.nodes.nfv.VduCp
  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.

6.8.11 tosca.nodes.nfv.VduSubCp

6.8.11.1 Description

A VduSubCp node type represents the Subport 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 through a trunk port. Table 6.8.11.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 6.8.11.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduSubCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:VduSubCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.VduSubCp</td>
</tr>
</tbody>
</table>

6.8.11.2 Properties

The properties of the VduSubCp node type shall comply with the provisions set out in table 6.8.11.2-1.

Table 6.8.11.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>segmentation_type</td>
<td>no</td>
<td>string</td>
<td>Allowed values:</td>
<td>Specifies the encapsulation type for the traffics coming in and out of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vlan, inherit</td>
<td>trunk subport.</td>
</tr>
<tr>
<td>segmentation_id</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Specifies the segmentation ID for the subport, which is used to differentiate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the traffics on different networks coming in and out of the trunk port.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If a value is provided here it may be overridden by a value provided at run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>time when the infrastructure does not support mapping of segmentation IDs.</td>
</tr>
</tbody>
</table>

6.8.11.3 Attributes

None.
6.8.11.4 Requirements

The requirements of the VduSubCp node type shall comply with the provisions set out in table 6.8.11.4-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trunk_binding</td>
<td>yes</td>
<td>tosca.capabilities.nfv.TrunkBindable</td>
<td></td>
<td>Describes the requirements for binding with trunk parent port.</td>
</tr>
</tbody>
</table>

6.8.11.5 Definition

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

```yaml
tosca.nodes.nfv.VduSubCp:
  derived_from: tosca.nodes.nfv.VduCp
  description: describes network connectivity between a VNFC instance based on this VDU and an internal VL through a trunk port
  properties:
    segmentation_type:
      type: string
      description: Specifies the encapsulation type for the traffics coming in and out of the trunk subport.
      required: false
      constraints:
        - valid_values: [ vlan, inherit ]
    segmentation_id:
      type: integer
      description: Specifies the segmentation ID for the subport, which is used to differentiate the traffics on different networks coming in and out of the trunk port.
      required: false
      constraints:
        - greater_or_equal: 0
    requirements:
      - trunk_binding:
        capability: tosca.capabilities.nfv.TrunkBindable
        relationship: tosca.relationships.nfv.TrunkBindsTo
        node: tosca.nodes.nfv.VduCp
        occurrences: [1, 1]
```

6.8.11.6 Example

See clause A.16.

6.8.11.7 Additional Requirements

The trunk_mode property of the VduSubCp node shall be set as false.

6.8.12 tosca.nodes.nfv.Vdu.OsContainer

6.8.12.1 Description

The Vdu.OsContainer node type represents the OsContainerDesc information element as defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.12.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].
6.8.12.2 Properties

The properties of the Vdu.OsContainer node type shall comply with the provisions set out in table 6.8.12.2-1.

### Table 6.8.12.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable name of the OS Container.</td>
</tr>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the OS Container.</td>
</tr>
<tr>
<td>requested_cpu_resources</td>
<td>no</td>
<td>integer</td>
<td>greater_than: 0</td>
<td>Number of CPU resources requested for the OS container (e.g. in milli-CPU-s).</td>
</tr>
<tr>
<td>cpu_resource_limit</td>
<td>no</td>
<td>integer</td>
<td>greater_than: 0</td>
<td>Number of CPU resources the OS container can maximally use (e.g. in milli-CPU).</td>
</tr>
<tr>
<td>requested_memory_resources</td>
<td>no</td>
<td>scalar-unit.size</td>
<td>greater_than: 0</td>
<td>Amount of memory resources requested for the OS container (e.g. in MB).</td>
</tr>
<tr>
<td>memory_resource_limit</td>
<td>no</td>
<td>scalar-unit.size</td>
<td>greater_than: 0</td>
<td>Amount of memory resources the OS container can maximally use (e.g. in MB).</td>
</tr>
<tr>
<td>requested_ephemeral_storage_resources</td>
<td>no</td>
<td>scalar-unit.size</td>
<td>greater_than: 0</td>
<td>Size of ephemeral storage resources requested for the OS container (e.g. in GB).</td>
</tr>
<tr>
<td>ephemeral_storage_resource_limit</td>
<td>no</td>
<td>scalar-unit.size</td>
<td>greater_than: 0</td>
<td>Size of ephemeral storage resources the OS container can maximally use (e.g. in GB).</td>
</tr>
<tr>
<td>extended_resource_requests</td>
<td>no</td>
<td>list of</td>
<td></td>
<td>Extended resources and their respective amount required by the container.</td>
</tr>
<tr>
<td>huge_pages_resources</td>
<td>no</td>
<td>list of</td>
<td></td>
<td>The requirement for huge pages resources. Each element in the list indicates a hugepage size and the total memory requested for hugepages of that size.</td>
</tr>
</tbody>
</table>

6.8.12.3 Attributes

None.

6.8.12.4 Capabilities

The capabilities of the Vdu.OsContainer node type shall comply with the provisions set out in table 6.8.12.4-1.

### Table 6.8.12.4-1: Capabilities

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>container_deployable</td>
<td>tosca.capabilities.nfv.ContainerDeployable</td>
<td></td>
<td>Defines ability of ContainerDeployable.</td>
</tr>
</tbody>
</table>
6.8.12.5 Definition

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

```yaml
tosca.nodes.nfv.Vdu.OsContainer:
  derived_from: tosca.nodes.Root
  description: Describes the resources of a single OS container within a VDU
  properties:
    name:
      type: string
      description: Human readable name of the OS container
      required: true
    description:
      type: string
      description: Human readable description of the OS container
      required: true
    requested_cpu_resources:
      type: integer
      description: Number of CPU resources requested for the OS container (e.g. in milli-CPU-s).
      required: false
      constraints:
        - greatest_or_equal: 0
    cpu_resource_limit:
      type: integer
      description: Number of CPU resources the OS container can maximally use (e.g. in milli-CPU).
      required: false
      constraints:
        - greatest_or_equal: 0
    requested_memory_resources:
      type: scalar-unit.size
      description: Amount of memory resources requested for the OS container (e.g. in MB).
      required: false
      memory_resource_limit:
        type: scalar-unit.size
        description: Amount of memory resources the OS container can maximum use (e.g. in MB).
        required: false
    requested_ephemeral_storage_resources:
      type: scalar-unit.size
      description: Size of ephemeral storage resources requested for the OS container (e.g. in GB).
      required: false
    ephemeral_storage_resource_limit:
      type: scalar-unit.size
      description: Size of ephemeral storage resources the OS container can maximum use (e.g. in GB).
      required: false
    extended_resource_requests:
      type: list
      description: Extended resources and their respective amount required by the container.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.ExtendedResourceData
        constraints:
          - min_length: 1
    huge_pages_resources:
      type: list
```

ETSII
description: The requirement for huge pages resources. Each element in the list indicates a hugepage size and the total memory requested for hugepages of that size.

required: false
type: tosca.datatypes.nfv.Hugepages
capabilities:
  container_deployable:
    type: tosca.capabilities.nfv.ContainerDeployable
    occurrences: [ 1, UNBOUNDED ]

6.8.12.6 Additional requirements

Node templates of type tosca.nodes.nfv.Vdu.OsContainer shall 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.OsContainer node template. The node template name of type tosca.nodes.nfv.Vdu.OsContainer 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.

6.8.13 tosca.nodes.nfv.Vdu.OsContainerDeployableUnit

6.8.13.1 Description

The Vdu.OsContainerDeployableUnit node type describes the aggregate of OS containers of a VDU (when realized as OS containers) which is a construct supporting the description of the deployment and operational behaviour of a VNFC.

Table 6.8.13.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Vdu.OsContainerDeployableUnit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:Vdu.OsContainerDeployableUnit</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.Vdu.OsContainerDeployableUnit</td>
</tr>
</tbody>
</table>

6.8.13.2 Properties

The properties of the Vdu.OsContainerDeployableUnit node type shall comply with the provisions set out in table 6.8.13.2-1.
Table 6.8.13.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td></td>
<td>Human readable name of the Vdu.</td>
</tr>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td></td>
<td>Human readable description of the Vdu.</td>
</tr>
<tr>
<td>logical_node</td>
<td>no</td>
<td>Map of tosca.data types.nfv.LogicalNodeData</td>
<td></td>
<td></td>
<td>Describes the Logical Node requirements.</td>
</tr>
<tr>
<td>requested_additional_capabilities</td>
<td>no</td>
<td>Map of tosca.data types.nfv.RequestedAdditionalCapabilities</td>
<td></td>
<td></td>
<td>Describes additional capability for a particular OS container.</td>
</tr>
<tr>
<td>nfvi_constraints</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>configurable_properties</td>
<td>no</td>
<td>map of tosca.data types.nfv.VnfcConfigurableProperties</td>
<td></td>
<td></td>
<td>Describes the configurable properties of all VNFC instances based on this VDU.</td>
</tr>
<tr>
<td>vdu_profile</td>
<td>yes</td>
<td>tosca.data types.nfv.VduProfile</td>
<td></td>
<td></td>
<td>Defines additional instantiation data for the VDU.OsContainerDeployableUnit node.</td>
</tr>
<tr>
<td>mcio_constraint_params</td>
<td>no</td>
<td>list of string</td>
<td>Valid values: See YAML definition constraints</td>
<td></td>
<td>Defines the parameter names for constraints expected to be assigned to MCIos realizing this Vdu.OsContainerDeployableUnit. The value specifies the standardized semantical context of the MIO constraints.</td>
</tr>
</tbody>
</table>

6.8.13.3 Attributes

None.

6.8.13.4 Requirements

The requirements of the Vdu.OsContainerDeployableUnit node type shall comply with the provisions set out in table 6.8.13.4-1.

Table 6.8.13.4-1: Requirements

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_storage</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualStorage</td>
<td></td>
<td>Describes storage requirements for a virtual_storage instance attached to the VDU realised by the Vdu.OsContainerDeployableUnit</td>
</tr>
<tr>
<td>container</td>
<td>yes</td>
<td>tosca.capabilities.nfv.ContainerDeployable</td>
<td></td>
<td>Describes the OsContainer requirements that encompass the Vdu.OsContainerDeployableUnit</td>
</tr>
</tbody>
</table>
6.8.13.5 Capabilities

The capabilities of the Vdu.OsContainerDeployableUnit node type shall comply with the provisions set out in table 6.8.13.5-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_binding</td>
<td>tosca.capabilities.nfv.VirtualBindable</td>
<td></td>
<td>Defines ability of VirtualBindable.</td>
</tr>
<tr>
<td>associable</td>
<td>tosca.capabilities.nfv.AssociableVdu</td>
<td></td>
<td>Defines ability of AssociableVDU.</td>
</tr>
</tbody>
</table>

6.8.13.6 Definition

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

```python
tosca.nodes.nfv.Vdu.OsContainerDeployableUnit:
    derived_from: tosca.nodes.Root
    description: Describes the aggregate of container of a VDU (when realized as OS containers) which 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
        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 OS container
            required: false
            entry_schema:
                type: tosca.datatypes.nfv.RequestedAdditionalCapability
        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
        vdu_profile:
            type: tosca.datatypes.nfv.VduProfile
```

# configurable_properties:
#    type: tosca.datatypes.nfv.VnfConfigurableProperties
#    required: false
# derived types are expected to introduce
# configurable_properties with its type derived from
# tosca.datatypes.nfv.VnfConfigurableProperties

vdu_profile:
    type: tosca.datatypes.nfv.VduProfile
description: Defines additional instantiation data for the Vdu.OsContainerDeployableUnit node
required: true
mcio_constraint_params:
  type: list
  entry_schema:
    type: string
  constraints:
    - valid_values:
      - affinity_nfvi_pop
      - affinity_zone
      - affinity_zone_group
      - affinity_nfvi_node
      - affinity_cis_node
      - anti_affinity_nfvi_pop
      - anti_affinity_zone
      - anti_affinity_zone_group
      - anti_affinity_nfvi_node
      - anti_affinity_cis_node
      - local_affinity_nfvi_pop
      - local_affinity_zone
      - local_affinity_zoneGroup
      - local_affinity_nfvi_node
      - local_affinity_cis_node
      - local_anti_affinity_nfvi_pop
      - local_anti_affinity_zone
      - local_anti_affinity_zone_group
      - local_anti_affinity_nfvi_node
      - local_anti_affinity_cis_node
      - node_additional_capability_ssd
      - node_additional_capability_dpdk
      - node_additional_capability_sriov
      - node_additional_capability_gpu
      - node_additional_capability_fpga
      - node_additional_capability_cpu_pin
      - node_capability_logical_numa
      - node_pool

description: Defines the parameter names for constraints expected to be assigned to MCIOs realizing this Vdu.OsContainerDeployableUnit. The value specifies the standardized semantical context of the MCIO constraints.
required: false
capabilities:
  virtual_binding:
    type: tosca.capabilities.nfv.VirtualBindable
    occurrences: [ 0, UNBOUNDED ]
  associable:
    type: tosca.capabilities.nfv.AssociableVdu
    occurrences: [ 1, 1 ]
requirements:
  - virtual_storage:
    capability: tosca.capabilities.nfv.VirtualStorage
    relationship: tosca.relationships.nfv.AttachesTo
    occurrences: [ 0, UNBOUNDED ]
  - container:
    capability: tosca.capabilities.nfv.ContainerDeployable
    relationship: tosca.relationships.nfv.DeploysTo
    occurrences: [ 1, UNBOUNDED ]
6.8.14  **tosca.nodes.nfv.Mciop**

### 6.8.14.1 Description

The Mciop node type does not correspond to an information element defined in ETSI GS NFV-IFA 011 [1]. It is a representation of the object described by the mciop artifact, capable of being profiled by the properties of the MciopProfile information element defined in ETSI GS NFV-IFA 011 [1]. Table 6.8.14.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Mciop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:Mciop</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.Mciop</td>
</tr>
</tbody>
</table>

### 6.8.14.2 Properties

None.

### 6.8.14.3 Attributes

None.

### 6.8.14.4 Requirements

The requirements of the Mciop node type shall comply with the provisions set out in table 6.8.14.4-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>associatedVdus</td>
<td>yes</td>
<td>tosca.capabilities.nfv.AssociableVdu</td>
<td></td>
<td>Specifies Vdu.OsContainerDeployableUnit nodes that are associated to the Mciop.</td>
</tr>
</tbody>
</table>

### 6.8.14.5 Capabilities

None.

### 6.8.14.6 Definition

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

```yaml
tosca.nodes.nfv.Mciop:
  derived_from: tosca.nodes.Root
  description: Representation of the object described by the mciop artifact, capable of being profiled by the properties of the MciopProfile information element defined in ETSI GS NFV-IFA 011 [1].
  requirements:
    - associatedVdu:
        capability: tosca.capabilities.nfv.AssociableVdu
        relationship: tosca.relationships.nfv.MciopAssociates
        node: tosca.nodes.nfv.Vdu.OsContainerDeployableUnit
        occurrences: [1, UNBOUNDED]
```
6.8.14.7 Additional Requirements

The dependency requirement as defined in TOSCA-Simple-Profile-YAML-v1.3 [20] may be used towards other Mciop nodes to express the order of deployment.

Node templates of type tosca.nodes.nfv.Mciop may contain an artifact definition of type tosca.artifacts.nfv.HelmChart. There shall be a maximum number of one such artifact definition in a tosca.nodes.nfv.Mciop node template.

6.8.14.8 Example

FFS.

6.8.15 tosca.nodes.nfv.VirtualCp

6.8.15.1 Description

A VirtualCp node type represents the VirtualCpd information element as defined in ETSI GS NFV-IFA 011 [1], which describes a requirement to create a virtual connection point allowing the access to a number of VNFC instances (based on their respective VDUs). Table 6.8.15.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.VirtualCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.VirtualCp</td>
</tr>
</tbody>
</table>

6.8.15.2 Properties

The properties of the VirtualCp node type shall comply with the provisions set out in table 6.8.15.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalServiceData</td>
<td>yes</td>
<td>List of tosca.datatypes.nfv.AdditionalServiceData</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.8.15.3 Attributes

None.

6.8.15.4 Requirements

The requirements of the VirtualCp node type shall comply with the provisions set out in table 6.8.15.4-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>yes</td>
<td>tosca.capabilities.Node</td>
<td></td>
<td>Target shall be Vdu.OsContainerDeployableUnit or Vdu.Compute</td>
</tr>
<tr>
<td>virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link</td>
</tr>
</tbody>
</table>

6.8.15.5 Definition

The syntax of the VirtualCp node type shall comply with the following definition:
tosca.nodes.nfv.VirtualCp:
  derived_from: tosca.nodes.nfv.Cp
description: Describes a virtual connection point allowing access to a number of
VNFC instances (based on their respective VDUs).
properties:
  additionalServiceData:
    type: list
    entry_schema:
      type: tosca.datatypes.nfv.AdditionalServiceData
      description: References the VDU(s) which implement this service
      required: true
requirements:
  - target:
    capability: tosca.capabilities.Node
      relationship: tosca.relationships.DependsOn
      occurrences: [ 1, UNBOUNDED ]
  - virtual_link:
    capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [0, 1]

6.8.15.6 Example

VirtCp1:
  type: tosca.nodes.nfv.VirtualCp
  properties:
    layer_protocols: [ tcp ]
    protocol:
      - address_data:
        - address_type: ip_address
        13_address_data:
          floating_ip_activated: true
          ip_address_assignment: false
          ip_address_type: ipv6
          number_of_ip_address: 1
          associated_layer_protocol: ipv6
        additionalServiceData:
          - portData:
            protocol: tcp
            port: 8001
            portConfigurable: false
          - name: ConfigService
            protocol: tcp
            port: 443
            portConfigurable: false
    requirements:
      - target: Vdu_1

Vdu_1:
  type: tosca.nodes.nfv.Vdu.OsContainerDeployableUnit
  properties:
    name: "SWLB"
    description: "Software Load Balancer"
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 4
    requirements:
      - container: Vdu_1_Container_1
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) or between the workloads being deployed based on different Mciops. 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>PlacementGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.PlacementGroup</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.groups.nfv.PlacementGroup</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the group</td>
</tr>
</tbody>
</table>

6.9.1.3 Definition

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

```yaml
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) or between the workloads being deployed based on different Mciops
  properties:
    description:
      type: string
      description: Human readable description of the group
      required: true
    members: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.Vdu.OsContainerDeployableUnit, tosca.nodes.nfv.VnfVirtualLink, tosca.nodes.nfv.Mciop ]
```

6.9.1.4 Additional Requirements

A group with type tosca.groups.nfv.PlacementGroup shall contain more than one member with the same node type 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstantiationLevels</td>
<td>tosca.nfv:InstantiationLevels</td>
<td>tosca.policies.nfv.InstantiationLevels</td>
</tr>
</tbody>
</table>

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

6.10.1.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>VduInstantiationLevels</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VduInstantiationLevels</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>levels</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.VduLevel</td>
<td>Describes the Vdu.Compute levels of resources that can be used to instantiate the VNF using this flavour.</td>
<td></td>
</tr>
</tbody>
</table>

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, tosca.nodes.nfv.Vdu.OsContainerDeployableUnit ]
```

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Qualified Name</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualLinkInstantiationLevels</td>
<td>tosca.policies.nfv.VirtualLinkInstantiationLevels</td>
<td>tosca.policies.nfv.VirtualLinkInstantiationLevels</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>levels</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.VirtualLinkBitrateLevel</td>
<td>- min_length: 1</td>
<td>Describes the virtual link levels of resources that can be used to instantiate the VNF using this flavour.</td>
</tr>
</tbody>
</table>

### 6.10.3.3  Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ScalingAspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscafv:ScalingAspects</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.ScalingAspects</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspects</td>
<td>yes</td>
<td>Map of tosca.datatypes.nfv.</td>
<td></td>
<td>Describe maximum scale level for total number of scaling steps that can be applied to a particular aspect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ScalingAspect</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.10.5.3 Definition

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

```yaml
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 Additional Requirements

A scaling aspect for which only one scaling delta is defined in VduScalingAspectDeltas and VirtualLinkBitrateScalingAspectDeltas policies is called a “uniform aspect”. In the case of “uniform aspect”, the step_deltas properties of tosca.datatypes.nfv.ScalingAspect is optional. If step_deltas is included, the value shall be a list of entries of step_deltas.

6.10.5.5 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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduScalingAspectDeltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanyf:VduScalingAspectDeltas</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VduScalingAspectDeltas</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Represents the scaling aspect to which this policy applies.</td>
</tr>
<tr>
<td>deltas</td>
<td>yes</td>
<td>map of toscad.datatypes.nfv.VduLevel</td>
<td></td>
<td>Describes the Vdu.Compute scaling deltas to be applied for every scaling steps of a particular aspect.</td>
</tr>
</tbody>
</table>

6.10.6.3  Definition

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

```yaml
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, tosca.nodes.nfv.Vdu.OsContainerDeployableUnit]

```

6.10.6.4  Additional Requirements

In the case of "uniform aspect", the deltas properties shall have only one entry.
6.10.6.5 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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualLinkBitrateScalingAspectDeltas</td>
<td>tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas</td>
<td>tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Represents the scaling aspect to which this policy applies.</td>
</tr>
<tr>
<td>deltas</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.VirtualLinkBitrateLevel</td>
<td></td>
<td>Describes the VnfVirtualLink scaling deltas to be applied for every scaling steps of a particular aspect.</td>
</tr>
</tbody>
</table>

6.10.7.3 Definition

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

```yaml
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 Additional Requirements

In the case of “uniform aspect”, the deltas properties shall have only one entry.

6.10.7.5 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VduInitialDelta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VduInitialDelta</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VduInitialDelta</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_delta:</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VduLevel</td>
<td></td>
<td>Represents the initial minimum size of the VNF.</td>
</tr>
</tbody>
</table>

6.10.8.3 Definition

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

```yaml
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,
tosca.nodes.nfv.Vdu.OsContainerDeployableUnit]
```

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinkBitrateInitialDelta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.VirtualLinkBitrateInitialDelta</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VirtualLinkBitrateInitialDelta</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial_delta:</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VirtualLinkBitrateLevel</td>
<td></td>
<td>Represents the initial minimum size of the VNF.</td>
</tr>
</tbody>
</table>

6.10.9.3  Definition

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

```yaml
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.Vdu.OsContainerDeployableUnit 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.Vdu.OsContainerDeployableUnit or tosca.nodes.nfv.VnfVirtualLink or tosca.nodes.nfv.Mciop
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) or between sets of virtualisation containers, realized by OS containers, to be created
based on different MCIOPs 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
</table>

Table 6.10.10.1-2: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AntiAffinityRule</td>
<td>tosca:nfv:AntiAffinityRule</td>
<td>tosca.policies.nfv.AntiAffinityRule</td>
</tr>
</tbody>
</table>

### 6.10.2 Properties

The properties of the AffinityRule and AntiAffinityRule types shall comply with the provisions set out in
table 6.10.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Yes</td>
<td>String</td>
<td>Possible values are &quot;nfvi_pop&quot;, &quot;zone&quot;, &quot;zone_group&quot;, &quot;nfvi_node&quot;, &quot;network_link_and_node&quot;, &quot;container_namespace&quot;, &quot;cis_node&quot;</td>
<td>Specifies the scope of the affinity or anti-affinity rule. See notes 1, 2 and 3.</td>
</tr>
<tr>
<td>nfvi_maintenance_info</td>
<td>no</td>
<td>tosca.datatypes.nfv.NfviMaintenanceInfo</td>
<td>Provides information on the impact tolerance and rules to be observed when a group of instances based on the same Vdu.Compute node (for VM based VDU) is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades). See notes 4 and 5.</td>
<td></td>
</tr>
</tbody>
</table>

NOTE 1: When used in an anti-affinity relationship, the "network_link_and_node" scope is conceptually similar to link and node disjoint paths capabilities used commonly in network Traffic Engineering (TE). For example, as in Fast Reroute Resource Reservation Protocol Traffic Engineering (RSVP-TE) for Label-Switched Path (LSP) tunnels as introduced in IETF RFC 4090 [i.17].

NOTE 2: The "container"namespace" is only applicable when the targets of the policy are exclusively nodes of type tosca.nodes.nfv.Mciop.

NOTE 3: The "cis-node" scope is only applicable when the targets of the policy are exclusively nodes of type tosca.nodes.nfv.Vdu.OsContainerDeployableUnit.

NOTE 4: The nfvi_maintenance_info property may only be present if there is only one node template with node type tosca.nodes.nfv.Vdu.Compute (for VM based VDU) set as the targets.

NOTE 5: An NFVI level operation (e.g. restart of a virtual machine) can impact a VNF and the VNF can be able to tolerate only a limited number of such impacts simultaneously. The nfvi_maintenance_group_info provides constraints related to the tolerated simultaneous impacts on a group of resources so that negative impact on VNF functionality can be avoided during NFVI maintenance operations.
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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>Yes</td>
<td>string[]</td>
<td>Valid types:</td>
<td>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.Vdu.OsContainerDeployableUnit 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.Vdu.OsContainerDeployableUnit and tosca.nodes.nfv.VnfVirtualLink and tosca.nodes.nfv.Mciop or a tosca.groups.nfv.PlacementGroup.</td>
</tr>
</tbody>
</table>

6.10.10.4 Definition

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

```yaml
tosca.policies.nfv.AffinityRule:
  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, network_link_and_node, container_namespace, cis_node ]
        - nfvi_maintenance_group_info:
          type: tosca.datatypes.nfv.NfviMaintenanceInfo
          description: Provides information on the impact tolerance and rules to be observed when a group of instances based on the same Vdu.Compute (for VM based VDU) node is impacted during NFVI operation and maintenance (e.g. NFVI resource upgrades).
          required: false
    targets: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink, tosca.groups.nfv.PlacementGroup, tosca.nodes.nfv.Mciop, tosca.nodes.nfv.Vdu.OsContainerDeployableUnit ]
```

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

```yaml
tosca.policies.nfv.AntiAffinityRule:
  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, 
  network_link_and_node, container_namespace, cis_node ]

  nfvi_maintenance_group_info:
  - type: tosca.datatypes.nfv.NfviMaintenanceInfo
  - description: Provides information on the impact tolerance and rules to be 
    observed when a group of instances based on the same Vdu.Compute node is impacted 
    during NFVI operation and maintenance (e.g. NFVI resource upgrades).
  - required: false
  - targets: [ tosca.nodes.nfv.Vdu.Compute, tosca.nodes.nfv.VnfVirtualLink, 
    tosca.groups.nfv.PlacementGroup, tosca.nodes.nfv.Mciop, 
    tosca.nodes.nfv.Vdu.OsContainerDeployableUnit ]

### 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupportedVnfInterface</td>
<td>tosca.policies.nfv.SupportedVnfInterface</td>
<td>tosca.policies.nfv.SupportedVnfInterface</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface_name</td>
<td>yes</td>
<td>string</td>
<td>Valid values</td>
<td>Identifies an interface produced by the VNF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vnf_indicator, vnf_configuration, vnf_lcm_coordinat ion</td>
<td></td>
</tr>
<tr>
<td>details</td>
<td>no</td>
<td>tosca.datatypes.nfv.InterfaceDetails</td>
<td>Provide additional data to access the interface endpoint (e.g. API URI prefix).</td>
<td></td>
</tr>
</tbody>
</table>

6.10.12.3  Definition

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

```yaml
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, vnf_lcm_coordination ]
  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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>SecurityGroupRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.SecurityGroupRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.SecurityGroupRule</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string[]</td>
<td>Valid types: tosca.nodes.nfv.VduCp, tosca.nodes.nfv.VnfExtCp, tosca.nodes.nfv.Vdu.OsContainerDeployableUnit</td>
<td>Target connection points of VduCp and/or VnfExtCp, or target VDUs based on one or set of OS containers. See note.</td>
</tr>
</tbody>
</table>

NOTE: If the OsContainerDeployableUnit is used as the target, the security rule is applicable for all the connection points related to the OsContainerDeployableUnit.

6.10.13.4 Definition

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

```yaml
tosca.policies.nfv.SecurityGroupRule:
derived_from: tosca.policies.nfv.Abstract.SecurityGroupRule
```
6.10.13.5 Additional Requirements

None.

6.10.14 tosca.policies.nfv.VnfIndicator

6.10.14.1 Description

The VnfIndicator is a base policy type for defining VNF indicator specific policies that define the conditions to assess and the action to perform when a VNF indicator changes value as defined in ETSI GS NFV-IFA 011 [1].

Table 6.10.14.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfIndicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.VnfIndicator</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VnfIndicator</td>
</tr>
</tbody>
</table>

6.10.14.2 Properties

The properties of the VnfIndicator policy type shall comply with the provisions set out in table 6.10.14.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>yes</td>
<td>string</td>
<td>valid values: See YAML definition constraints</td>
<td>Describe the source of the indicator.</td>
</tr>
</tbody>
</table>

6.10.14.3 Definition

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

```yaml
tosca.policies.nfv.VnfIndicator:
  derived_from: tosca.policies.Root
  description: The VnfIndicator policy type is a base policy type for defining VNF indicator specific policies that define the conditions to assess and the action to perform when a VNF indicator changes value as defined in ETSI GS NFV-IFA 011 [1].
  properties:
    source:
      type: string
      description: Describe the source of the indicator.
      required: true
      constraints:
        - valid_values: [ vnf, em, both_vnf_and_em ]
  targets: [ tosca.nodes.nfv.VNF ]
```
6.10.14.4 Additional requirements

The VNFD service template may include VNF indicator specific policies of VnfIndicator type with the following requirements:

a) it shall include one or more trigger definitions which:
   - shall include an event with a value equal to the full name of a notification in the VnfIndicator interface definition of the VNF node where the policy applies;
   - may include a condition definition which can assert the value of vnf indicator attributes and other node attributes using arbitrary AND and OR combinations of the individual assertions;
   - may include an action invoking one or multiple operations of the Vnflcm interface;

b) the target shall be set to the node template to which the policy applies, i.e. to the node template of the VNF specific type present in the topology template that represents a particular deployment flavour.

6.10.15 tosca.policies.nfv.VnfPackageChange

6.10.15.1 Description

The VnfPackageChange type is a policy type specifying the processes and rules to be used for performing the resource related tasks, to change VNF instance to a different VNF Package (destination package) as defined in ETSI GS NFV-IFA 011 [1]. Table 6.10.15.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfPackageChange</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VnfPackageChange</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VnfPackageChange</td>
</tr>
</tbody>
</table>

6.10.15.2 Properties

The properties of the VnfPackageChange policy type shall comply with the provisions set out in table 6.10.15.2-1.
Table 6.10.15.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectors</td>
<td>yes</td>
<td>list of tosca.datatypes.nfv.VnfPackageChangeSelector</td>
<td></td>
<td>Information to identify the source and destination VNFD for the change, and the related deployment flavours. See note 1.</td>
</tr>
<tr>
<td>modification_qualifier</td>
<td>yes</td>
<td>string</td>
<td>up, down</td>
<td>Specifies the type of modification resulting from transitioning from srcVnfdId to dstVnfdId. The possible values are UP indicating that the destination VNF version is newer than the source version, DOWN indicating that the destination VNF version is older than the source version.</td>
</tr>
<tr>
<td>additional_modification_description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Additional information to qualify further the change between the two versions.</td>
</tr>
<tr>
<td>component_mappings</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.VnfPackageChangeComponentMapping</td>
<td></td>
<td>Mapping information related to identifiers of components in source VNFD and destination VNFD that concern to the change process.</td>
</tr>
<tr>
<td>destination_flavour_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the deployment flavour in the destination VNF package for which this change applies. The flavour ID is defined in the destination VNF package.</td>
</tr>
<tr>
<td>actions</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>List of applicable supported LCM coordination action names specified in this VNFD (action_name) as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction.</td>
</tr>
<tr>
<td>referenced_coordination_actions</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>List of names of coordination actions not specified within this VNFD as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction. See note 2.</td>
</tr>
</tbody>
</table>

NOTE 1: If selectors includes multiple entries, all other properties apart from the selectors in the VnfPackageChange policy define the package change that is applicable to any change path defined by any of the entries in the selectors. If change paths require e.g. different component_mappings they shall be described by different VnfPackageChange policies. Each triplet (source_descriptor_id, destination_descriptor_id, source_flavour_id) represented by one entry in the selectors shall not appear more than once in a VNFD. If a triplet occurs in both, source and destination package, the content of the identified VnfPackageChange policies (apart from the selectors) shall be the same.

NOTE 2: Naming conventions for coordination names are defined in names as specified in clause 10.7 of ETSI GS NFV-SOL 002 [22].

6.10.15.3 Definition

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

```yaml
tosca.policies.nfv.VnfPackageChange:
  derived_from: tosca.policies.Root
  description: policy type specifying the processes and rules to be used for performing the resource related tasks, to change VNF instance to a different VNF Package (destination package)
  properties:
    selector:
      type: list
```
entry_schema:
  type: tosca.datatypes.nfv.VnfPackageChangeSelector
  description: Information to identify the source and destination VNFD for
  the change, and the related deployment flavours.
  required: true
  constraints:
    - min_length: 1
  modification_qualifier:
    type: string
    description: Specifies the type of modification resulting from
    transitioning from srcVnfdId to dstVnfdId. The possible values are UP indicating
    that the destination VNF version is newer than the source version, DOWN indicating
    that the destination VNF version is older than the source version.
    constraints: [ valid_values: [ up, down ] ]
    required: true
  additional_modification_description:
    type: string
    description: Additional information to qualify further the change between
    the two versions.
    required: false
  component_mappings:
    type: list
    entry_schema:
      type: tosca.datatypes.nfv.VnfPackageChangeComponentMapping
      description: Mapping information related to identifiers of components in
      source VNFD and destination VNFD that concern to the change process.
      required: false
  destination_flavour_id:
    type: string
    description: Identifies the deployment flavour in the destination VNF
    package for which this change applies. The flavour ID is defined in the
    destination VNF package.
    required: true
  actions:
    type: list
    description: List of applicable supported LCM coordination action names
    (action_name) specified in this VNFD as a TOSCA policy of a type derived from
    tosca.policies.nfv.LcmCoordinationAction.
    required: false
    entry_schema:
      type: string
      referenced_coordination_actions:
        type: list
        description: List of names of coordination actions not specified within
        this VNFD as a TOSCA policy of a type derived from
tosca.policies.nfv.LcmCoordinationAction.
        required: false
        entry_schema:
        type: string
        targets: [ tosca.nodes.nfv.VNF ]

6.10.15.4 Additional Requirements

The VnfPackageChange specific type policy shall have exactly one trigger with an event and an action.

The event value shall be set to change_current_package_notification notification of Vnflcm interface.

The target shall be set to the node template to which the policy applies, i.e. to the node template of the VNF specific
type present in the topology template that represents a particular deployment flavour.
The action value shall be set to either change_current_package operation on the Vnflcm (in case the same LCM script or no LCM script with the same set of "additionalParams" or no "additionalParams" is suitable for all change paths) or one of the VNF-specific operations on the ChangeCurrentVnfPackage interface (in case different change paths require different LCM scripts or no LCM script potentially with different sets of "additionalParams" or no "additionalParams").

The policy shall be applied when the actual values of the descriptor_id and flavour_id of the source VNF type and the descriptor_id of the destination VNF type match the source_descriptor_id, destination_descriptor_id and source_flavour_id properties, respectively, of the selector in the policy.

VNF-specific coordination actions shall be declared with their parameters in data types derived from tosca.datatypes.nfv.VnfLcmOpCoord (see clause 6.2.67), and a interface_name property of the tosca.policies.nfv.SupportedVnfInterface set to "vnf lcm Coordination" shall be specified in the related deployment flavour to signal that this interface is exposed by the VNF.

NOTE: During and after the VNF Package change all information related to any other deployment flavours in the source VNFD than the source flavour is no longer applicable.

6.10.15.5 Example

The following example template illustrates a VNFD which is supporting VNF package change info via using VnfPackageChange policy, additional_parameters input for each case of the selector and VNF specific interface (tosca.interfaces.nfv.ChangeCurrentVnfPackage), LCM coordination actions, and other data types. A policy definition of change_from_version_1 and change_to_version_1 has a trigger. The action will be taken when the conditions of the selector values are satisfied. The policy has an event associated to the notification of tosca.interfaces.nfv.Vnflcm.change_current_package_notification.

tosca_definitions_version: tosca_simple_yaml_1_3
imports: ..
node_types:
  MyCompany.SunshineDB.2_0.2_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
        type: string
        constraints: [ equal: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5 ]
        default: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: SunshineDB ]
        default: SunshineDB
      software_version:
        type: string
        constraints: [ equal: '2.0' ]
        default: '2.0'
      descriptor_version:
        type: string
        constraints: [ equal: '2.0' ]
        default: '2.0'
      flavour_id:
        type: string
        constraints: [ equal: default ]
        default: default
    interfaces:
      MyCompanyChangeCurrentVnfPackage:
type: MyCompany.interfaces.nfv.ChangeCurrentVnfPackage

interface_types:
  MyCompany.interfaces.nfv.ChangeCurrentVnfPackage
    derived_from: tosca.interfaces.nfv.ChangeCurrentVnfPackage
  operations:
    change_from_version_1:
      description: operation for change from version 1 to 2
      inputs:
        additional_parameters:
          type:
            MyCompany.datatypes.nfv.VnfChangeFromVersion1AdditionalParameters
    change_to_version_1:
      description: operation for change from version 2 to 1
      inputs:
        additional_parameters:
          type: MyCompany.datatypes.nfv.VnfChangeToVersion1AdditionalParameters

data_types:
  ...

MyCompany.datatypes.nfv.VnfChangeFromVersion1AdditionalParameters:
  derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  properties:
    parameter_1:
      type: string
    parameter_2:
      type: string

MyCompany.datatypes.nfv.VnfChangeToVersion1AdditionalParameters:
  derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
  properties:
    parameter_3:
      type: string
    parameter_4:
      type: string

MyCompany.datatypes.nfv.VnfChangeFromVersion1OpCoord:
  derived_from: tosca.datatypes.nfv.VnfLcmOpCoord
  properties:
    description:
      type: string
    required: true
    constraints: [ equal: 'MyCompany vnf change from version 1 operation coordination information' ]
    default: 'MyCompany vnf change from version 1 operation coordination information'
    endpoint_type:
      type: string
    required: true
    constraints: [ equal: 'vnf' ]
    default: 'vnf'
    coordination_stage:
      type: string
    required: true
    constraints: [ equal: 'start' ]
    default: 'start'
    input_parameters:
      type: MyCompany.datatypes.nfv.VnfChangeFromVersion1InputOpCoordParams
    required: true
    output_parameters:
type: MyCompany.datatypes.nfv.VnfChangeGeneralOutputOpCoordParams
required: true

MyCompany.datatypes.nfv.VnfChangeToVersion1OpCoord:
derived_from: tosca.datatypes.nfv.VnfLcmOpCoord
properties:
description:
type: string
required: true
constraints: [ equal: 'MyCompany vnf change to version 1 operation coordination information' ]
default: 'MyCompany vnf change to version 1 operation coordination information'
endpoint_type:
type: string
required: true
constraints: [ equal: 'vnf' ]
default: 'vnf'
coordination_stage:
type: string
required: true
constraints: [ equal: 'start' ]
default: 'start'
input_parameters:
type: MyCompany.datatypes.nfv.VnfChangeToVersion1InputOpCoordParams
required: true
output_parameters:
type: MyCompany.datatypes.nfv.VnfChangeGeneralOutputOpCoordParams
required: true

MyCompany.datatypes.nfv.InstantiateOpCoord:
derived_from: tosca.datatypes.nfv.VnfLcmOpCoord
properties:
description:
type: string
required: true
constraints: [ equal: 'MyCompany vnf instantiate operation coordination information' ]
default: 'MyCompany vnf instantiate operation coordination information'
endpoint_type:
type: string
required: true
constraints: [equal: 'vnf' ]
default: 'vnf'
coordination_stage:
type: string
required: true
constraints: [ equal: 'start' ]
default: 'start'
input_parameters:
type: MyCompany.datatypes.nfv.InstantiateInputOpCoordParams
required: true

MyCompany.datatypes.nfv.InstantiateInputOpCoordParams:
derived_from: tosca.datatypes.nfv.InputOpCoordParams
properties:
data1:
type: string
required: true
constraints: [ equal: 'value_1' ]
default: 'value_1'
MyCompany.datatypes.nfv.VnfChangeFromVersion1InputOpCoordParams:
    derived_from: tosca.datatypes.nfv.InputOpCoordParams
    properties:
        data2:
            type: string
            required: true
            constraints: [ equal: 'value_2' ]
            default: 'value_2'

MyCompany.datatypes.nfv.VnfChangeToVersion1InputOpCoordParams:
    derived_from: tosca.datatypes.nfv.InputOpCoordParams
    properties:
        data3:
            type: string
            required: true
            constraints: [ equal: 'value_3' ]
            default: 'value_3'

MyCompany.datatypes.nfv.VnfChangeGeneralOutputOpCoordParams:
    derived_from: tosca.datatypes.nfv.OutputOpCoordParams
    properties:
        data4:
            type: string
            required: true
            constraints: [ equal: 'value_4' ]
            default: 'value_4'

..

deploy_policies:
    tosca.policies.nfv.LcmCoordinationAction.InstantiateHelloHandshake:
        derived_from: tosca.policies.nfv.LcmCoordinationAction
        properties:
            action:
                type: MyCompany.datatypes.nfv.InstantiateOpCoord

tosca.policies.nfv.VnfChangeFromVersion1Action1OpCoord:
    derived_from: tosca.policies.nfv.LcmCoordinationAction
    properties:
        action:
            type: MyCompany.datatypes.nfv.VnfChangeFromVersion1OpCoord

tosca.policies.nfv.VnfChangeToVersion1Action1OpCoord:
    derived_from: tosca.policies.nfv.LcmCoordinationAction
    properties:
        action:
            type: MyCompany.datatypes.nfv.VnfChangeToVersion1OpCoord

topology_template:
    substitution_mappings:
        node_type: MyCompany.SunshineDB.2_0.2_0
    ...

node_templates:
    sunshine_db:
        type: MyCompany.SunshineDB.2_0.2_0
    ...
    interfaces:
        MyCompanyChangeCurrentVnfPackage:
operations:
  ...
  change_from_version_1:
    implementation: change-from-version-1.workbook.mistral.yaml
  change_to_version_1:
    implementation: change-to-version-1.workbook.mistral.yaml

server:
  type: MyCompany.nodes.nfv.Vdu.Aux
  ...

volume:
  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
  ...

server_internal_cp:
  type: tosca.nodes.nfv.VduCp
  ...

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

policies:
  ...
  - instantiation_levels:
      type: tosca.policies.nfv.InstantiationLevels
        properties:
          levels:
            single:
              description: ..
            quadruple:
              description: ..
            default_level: single
  ...
  - change_from_version_1_0 and 1_1:
      type: tosca.policies.nfv.VnfPackageChange
      properties:
        selectors:
          - source_descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
            destination_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
            source_flavour_id: simple
            - source_descriptor_id: a9c82e92-0007-11ec-9a03-0242ac130003
              destination_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
              source_flavour_id: simple
              modification_qualifier: up
              additional_modification_description:..
        component_mappings:
          - component_type: vdu
            source_id: dbBackend
            destination_id: server
            description:..
          - component_type: virtual_storage
            source_id: mariaDbStorage
            destination_id: volume
            description:..
          - component_type: instantiation_level
            source_id: single
destination_id: default
description: ..

destination_flavour_id: default
actions:
- vnd.mycompany.VnfChangeFromVersion1Action1
triggers:
  change_from_version_1:
  event:
tosca.interfaces.nfv.Vnflcm.change_current_package_notification
  action:
    - call_operation:
        MyCompanyChangeCurrentVnfPackage.change_from_version_1
targets: [sunshine_db]

- change_to_version_1
  type: tosca.policies.nfv.VnfPackageChange
  properties:
    selectors:
      - source_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
        destination_descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
        source_flavour_id: default
        modification_qualifier: down
        additional_modification_description: ..
    component_mappings:
      - component_type: vdu
        source_id: server
        destination_id: dbBackend
        description: ..
      - component_type: virtual_storage
        source_id: volume
        destination_id: mariaDbStorage
        description:..
      - component_type: instantiation_level
        source_id: default
        destination_id: single
        description:..

destination_flavour_id: simple
actions:
- vnd.mycompany.VnfChangeToVersion1Action1
triggers:
  change_to_version_1:
  event:
tosca.interfaces.nfv.Vnflcm.change_current_package_notification
  action:
    - call_operation:
        MyCompanyChangeCurrentVnfPackage.change_to_version_1
targets: [sunshine_db]

- instantiate_HelloHandshake:
  type: tosca.policies.nfv.LcmCoordinationAction.InstantiateHelloHandshake
  properties:
    action_name: vnd.mycompany.hellohandshake

- instantiate_OpCoord:
  type: tosca.policies.nfv.LcmCoordinationsForLcmOperation
  properties:
    vnf_lcm_operation: instantiate
    actions:
      - vnd.mycompany.hellohandshake
- terminate_OpCoord:
  type: tosca.policies.nfv.LcmCoordinationsForLcmOperation
  properties:
    vnf_lcm_operation: terminate
    referenced_coordination_actions:
      - etsi.nfv.take-vnf-out-of-service

- VnfChangeFromVersion1Action1_OpCoord:
  type: tosca.policies.nfv.LcmCoordinationAction
  properties:
    action_name: vnd.mycompany.VnfChangeFromVersion1Action1

- VnfChangeToVersion1Action1_OpCoord:
  type: tosca.policies.nfv.LcmCoordinationAction
  properties:
    action_name: vnd.mycompany.VnfChangeToVersion1Action1

The following example template illustrates a VNFD which is supporting VNF package change info via using VnfPackageChange policy, without additional_parameters input and Vnflcm interface operation. A policy definition of change_from_version_1 and change_to_version_1 has a trigger. The action will be taken when the conditions of the selector values are satisfied. The policy has an event associated to the notification of tosca.interfaces.nfv.Vnflcm.change_current_package_notification.

tosca_definitions_version: tosca_simple_yaml_1_3
imports: ..
node_types:
  MyCompany.SunshineDB.2_0.2_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
        type: string
        constraints: [ equal: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5 ]
        default: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: SunshineDB ]
        default: SunshineDB
      software_version:
        type: string
        constraints: [ equal: '2.0' ]
        default: '2.0'
      descriptor_version:
        type: string
        constraints: [ equal: '2.0' ]
        default: '2.0'
      flavour_id:
        type: string
        constraints: [ equal: default ]
        default: default
  ...
interfaces:
MyCompanyVnflcm:
  type: tosca.interfaces.nfv.Vnflcm

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.2_0.2_0

node_templates:
  sunshine_db:
    type: MyCompany.SunshineDB.2_0.2_0
    interfaces:
      Vnflcm:
        operations:
          ..
          change_current_package:

server:
  type: MyCompany.nodes.nfv.Vdu.Aux

volume:
  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage

server_internal_cp:
  type: tosca.nodes.nfv.VduCp

internal_vl:
  type: tosca.nodes.nfv.VnfVirtualLink

policies:
  ..
  - instantiation_levels:
      type: tosca.policies.nfv.InstantiationLevels
      properties:
        levels:
          single:
            description: ..
          quadruple:
            description: ..
        default_level: single

  - change_from_version_1_0 and 1_1:
      type: tosca.policies.nfv.VnfPackageChange
      properties:
        selectors:
          - source_descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
destination_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
source_flavour_id: simple
          - source_descriptor_id: a9c82e92-0007-11ec-9a03-0242ac130003
destination_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
source_flavour_id: simple
modification_qualifier: up
additional_modification_description: ..
**component_mappings:**
- **component_type:** vdu
  source_id: dbBackend
  destination_id: server
  description: ...
- **component_type:** virtual_storage
  source_id: mariaDbStorage
  destination_id: volume
  description: ...
- **component_type:** instantiation_level
  source_id: single
  destination_id: default
  description: ...

**destination_flavour_id:** default

**triggers:**
- **change_from_version_1:**
  event: tosca.interfaces.nfv.Vnflcm.change_current_package_notification
  action:
  - call_operation: Vnflcm.change_current_package
  targets: [ sunshine_db ]

- **change_to_version_1_0:**
  type: tosca.policies.nfv.VnfPackageChange
  properties:
  selectors:
  - source_descriptor_id: ebc68e34-0cfa-40ba-8b45-9caa31f9dcb5
    destination_descriptor_id: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
    source_flavour_id: default
    modification_qualifier: down
    additional_modification_description:..
  component_mappings:
  - **component_type:** vdu
    source_id: server
    destination_id: dbBackend
    description: ...
  - **component_type:** virtual_storage
    source_id: volume
    destination_id: mariaDbStorage
    description: ...
  - **component_type:** instantiation_level
    source_id: default
    destination_id: single
    description: ...

**destination_flavour_id:** simple

**triggers:**
- **change_to_version_1:**
  event: tosca.interfaces.nfv.Vnflcm.change_current_package_notification
  action:
  - call_operation: Vnflcm.change_current_package
  targets: [ sunshine_db ]
6.10.16  
tosca.policies.nfv.LcmCoordinationAction

6.10.16.1  Description

The LcmCoordinationAction type is a base type for deriving policy types which describe the LCM coordination actions supported by a VNF and/or expected to be supported by its EM. This policy concerns the whole VNF (deployment flavour) represented by the topology_template and thus has no explicit target list. Table 6.10.16.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>LcmCoordinationAction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.LcmCoordinationAction</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.LcmCoordinationAction</td>
</tr>
</tbody>
</table>

6.10.16.2  Properties

The properties of the LcmCoordinationAction policy type shall comply with the provisions set out in table 6.10.16.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Coordination action name. See note 1</td>
</tr>
<tr>
<td>action</td>
<td>yes</td>
<td>tosca.datatypes.nfv.VnfLcmOpCoord</td>
<td></td>
<td>Describes a set of information needed for coordination action in the VNF LCM operation. See note 2</td>
</tr>
</tbody>
</table>

NOTE 1: The action_name shall comply with naming conventions as specified in clause 10.7 of ETSI GS NFV-SOL 002 [22].
NOTE 2: Represents a place holder for specifying actions of a VNF-specific type derived from tosca.datatypes.nfv.VnfLcmOpCoord.

6.10.16.3  Definition

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

```yaml
tosca.policies.nfv.LcmCoordinationAction:
  derived_from: tosca.policies.Root
  description: The LcmCoordinationAction type is a policy type representing the LCM coordination actions supported by a VNF and/or expected to be supported by its EM for a particular VNF LCM operation. This policy concerns the whole VNF (deployment flavour) represented by the topology_template and thus has no explicit target list.
  properties:
    action_name:
      type: string
      description: Coordination action name.
      required: true
      # action: #represents a place holder for specifying actions of a VNF-specific type derived from tosca.datatypes.nfv.VnfLcmOpCoord
      # type: tosca.datatypes.nfv.VnfLcmOpCoord
      # description: Describes a set of information needed for coordination action in the VNF LCM operation.
      # required: true
```
6.10.16.4 Additional Requirements

A LcmCoordinationAction policy shall contain only one LCM coordination action. The VNFD shall define a policy type derived from tosca.policies.nfv.LcmCoordinationAction for each of VNF LCM coordination action. The LCM coordination action specific derived policy type shall contain a property ‘action’ of a datatype derived from tosca.datatypes.nfv.VnfLcmCoord.

The interface_name property of the tosca.policies.nfv.SuppliedVnflInterface set to "vnf lcm coordination" shall be specified in the related deployment flavour to signal that this interface is exposed by the VNF.

6.10.17 tosca.policies.nfv.LcmCoordinationsForLcmOperation

6.10.17.1 Description

The LcmCoordinationsForLcmOperation type is a policy type representing supported LCM coordination actions associated to a VNF LCM operation. This policy concerns the whole VNF (deployment flavour) represented by the topology_template and thus has no explicit target list. Table 6.10.17.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>LcmCoordinationsForLcmOperation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.LcmCoordinationsForLcmOperation</td>
</tr>
</tbody>
</table>

6.10.17.2 Properties

The properties of the LcmCoordinationsForLcmOperation policy type shall comply with the provisions set out in table 6.10.17.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnf_lcm_operation</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>The VNF LCM operation the LCM coordination actions are associated with.</td>
</tr>
<tr>
<td>actions</td>
<td>no</td>
<td>list of string</td>
<td>List of applicable supported LCM coordination action names (action_name) specified in this VNFD as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction. See note 1 and note 2.</td>
<td></td>
</tr>
<tr>
<td>referenced_coordination_actions</td>
<td>no</td>
<td>list of string</td>
<td>List of names of coordination actions not specified within this VNFD as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction. See note 1 and note 2.</td>
<td></td>
</tr>
</tbody>
</table>

NOTE 1: At least one of the actions and referenced_coordination_actions properties shall be present.
NOTE 2: Naming conventions for coordination names are defined in names as specified in clause 10.7 of ETSI GS NFV-SOL 002 [22].
6.10.17.3 Definition

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

```yaml
tosca.policies.nfv.LcmCoordinationsForLcmOperation:
derived_from: tosca.policies.Root
description: The LcmCoordinationsForLcmOperation type is a policy type representing supported LCM coordination actions associated to a VNF LCM operation. This policy concerns the whole VNF (deployment flavour) represented by the topology_template and thus has no explicit target list.
properties:
  vnf_lcm_operation:
    type: string
    description: The VNF LCM operation the LCM coordination actions are associated with.
    required: true
    constraints:
    - valid_values: [instantiate, scale, scale_to_level, change_flavour, terminate, heal, operate, change_ext_conn, modify_info, create_snapshot, revert_to_snapshot ]
  actions:
    type: list
    description: List of applicable supported LCM coordination action names (action_name) specified in this VNFD as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction.
    required: false
    entry_schema:
      type: string
  referenced_coordination_actions:
    type: list
    description: List of names of coordination actions not specified within this VNFD as a TOSCA policy of a type derived from tosca.policies.nfv.LcmCoordinationAction.
    required: false
    entry_schema:
      type: string
```

6.10.17.4 Additional Requirements

The VNFD may define one LcmCoordinationsForLcmOperation policy per each VNF LCM operation (vnf_lcm_operation).

VNF lifecycle management coordination actions are invoked by the VNFM towards the VNF instance or towards operation supporting management systems (e.g., EM). They can be standardized or VNF-specific. To distinguish between both categories, clause 10.7 of ETSI GS NFV-SOL 002 [22] defines namespaces for the values of the coordination action names.

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 two levels of service templates

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 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. The file names of all the lower service templates shall be declared as the value of the Other-Definitions key as specified in TOSCA-Simple-Profile-YAML-v1.3 [20] in the TOSCA.meta file of the VNF package.

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-v1.3 [20] and shall include:

a) an import statement referencing the TOSCA types definition file as defined in clause B.2;

b) an import statement referencing a yaml file which contains a VNF specific node type definition;

c) optionally, import statements referencing additional VNF-specific files containing only type definitions used by this TOSCA service template; and

d) a topology template with a node template of the VNF specific node type, which:

- shall include the flavour_id and other properties that are marked as required but do not have a default value in the VNF specific node type definition;
- shall include the requirements as defined in clause 6.8.1;
- may include other properties specified in the VNF specific node type definition, excluding the vnf_profile property;
- may include a substitute directive;

e) optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

Irrespective of the presence of absence of the substitute directive, the deployment and lifecycle management of instances of this VNF node type shall be done by means of substitution by any of the lower level service templates as declared in the Other-Definitions of the TOSCA.meta file in the VNF package. The VNFD consumer shall silently ignore the substitute directive if explicit directives are not supported.

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-v1.3 [20] and shall include:

a) an import statement referencing the TOSCA types definition file as defined in clause B.2;

b) an import statement referencing a yaml file which contains a VNF specific node type definition which shall be derived from tosca.nodes.nfv.VNF node type as defined in clause 6.8.1;

c) optionally, additional VNF-specific type definitions and 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 node type as defined in the VNF specific node type definition service template;
  - a flavour_id property and its value as defined in substitution_filter which identifies the DF corresponding to this low level template within the VNFD;
NOTE 1: Starting with version 3.3.1 of the present document, the property_mapping grammar was changed to support substitution_filter. For backward compatibility, TOSCA-Simple-Profile-YAML-v1.3 [20], clause 3.8.8.3 specifies the provisions for handling the previous grammar. Support of the Release 2 property_mapping grammar can be removed in subsequent versions of the present document.

- 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, the dependency requirements as defined in TOSCA-Simple-Profile-YAML-v1.3 [20] may be used between different node templates of type Vdu.Compute (or a derived node type) to specify the order in which instances of the VNFCs have to be created;
- additional group definitions, policy definitions and parameter definitions if applicable;

e) optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

NOTE 2: The format and structure of a VNF package is defined in ETSI GS NFV-SOL 004 [i.6].

NOTE 3: 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 filter for selecting a particular lower level TOSCA service template inside the VNF package as described in TOSCA-Simple-Profile-YAML-v1.3 [20].

6.11.3 Single deployment flavour design with one service template

In case of the single deployment flavour scenario with one service template design, the VNFD shall use TOSCA-Simple-Profile-YAML-v1.3 [20] and shall include:

a) an import statement referencing the TOSCA types definition file as defined in clause B.2;

b) 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;

c) optionally, additional VNF-specific type definitions and 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, the dependency requirements as defined in TOSCA-Simple-Profile-YAML-v1.3 [20] may be used between different node templates of type Vdu.Compute (or a derived node type) to specify the order in which instances of the VNFCs have to be created;
- additional group definitions, policy definitions and parameter definitions if applicable;

e) optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

See clause A.5 for an example of VNFD design with single deployment flavour.
NOTE 1: 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 2: All the imported type definition files as indicated in the service template are considered as parts of a VNFD.

6.11.4 Package change (handling the Change current VNF Package request)

The Change current VNF Package operation, defined in ETSI GS NFV-SOL 002 [22] and ETSI GS NFV-SOL 003 [i.9], enables the NFVO to request the VNFM to change the current VNF Package, i.e. the VNF package on which a VNF instance is based, hence enabling VNF software modification. For such a request to be handled properly by the VNFM, the VNFD provides a policy of type tosca.policies.nfv.VnfPackageChange with the following characteristics:

- there may be as many instances of this policy type in the VNFD as change paths are supported;
- either the source or the destination of a change path shall be the current VNFD i.e. the one where the policy instance is present;
- a given instance of this policy type is triggered when the combination of the source_descriptor_id, destination_descriptor_id, source_flavour_id values in its selector property matches with the following:
  - source_descriptor_id: "vnfdId": attribute in VnfInstance
  - destination_descriptor_id: "vnfdId" attribute in ChangeCurrentVnfPkgRequest
  - source_flavour_id: "flavour" attribute in VnfInstance
- the operation designated by the trigger action in the policy definition is invoked upon receiving the Change current VNF Package request on the API (ETSI GS NFV-SOL 003 [i.9] or ETSI GS NFV-SOL 002 [22]);
- if "additionalParams" are to be received in the Change current VNF Package request on the API, then the input signature (additional_parameters) of the designated operation defines what "additionalParams" can be submitted as part of the operation request.

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-Profile-YAML-v1.3 [20], 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.
Table 7.1-1: Mapping of ETSI GS NFV-IFA 014 [2] information elements with TOSCA types

<table>
<thead>
<tr>
<th>ETSI NFV Information Element ETSI GS NFV-IFA 014 [2]</th>
<th>TOSCA type</th>
<th>Derived from</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSD</td>
<td>tosca.nodes.nfv.NS</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>Sapd</td>
<td>tosca.nodes.nfv.Sap</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>NsVirtualLinkDesc</td>
<td>tosca.nodes.nfv.NsVirtualLink</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>Pnfd</td>
<td>tosca.nodes.nfv.PNF</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>Vnfd</td>
<td>tosca.nodes.nfv.VNF</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>Vnffgd</td>
<td>tosca.groups.nfv.VNFFG</td>
<td>tosca.groups.Root</td>
</tr>
</tbody>
</table>

7.2 Data Types

7.2.1 Void

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-v1.3 [20].

Table 7.2.3.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NsVlProfile</td>
<td>toscanfv:NsVlProfile</td>
<td>tosca.datatypes.nfv.NsVlProfile</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_bitrate_requirements</td>
<td>yes</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td></td>
<td>Specifies the maximum bitrate requirements for a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>min_bitrate_requirements</td>
<td>yes</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td></td>
<td>Specifies the minimum bitrate requirements for a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>qos</td>
<td>no</td>
<td>tosca.datatypes.nfv.NsVirtualLinkQos</td>
<td></td>
<td>Specifies the QoS requirements of a VL instantiated according to this profile.</td>
</tr>
<tr>
<td>service_availability_level</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 1</td>
<td>If present, specifies the service availability level for the VL instance created from this profile. See note 2.</td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Type</td>
<td>Constraints</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>virtual_link_protocol_data</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.NsVirtualLinkProtocolData</td>
<td>1. greater_or_equal: 1</td>
<td>Specifies the protocol data for a virtual link. If more than 1 values are present, the order shall be the same as the order of the layer_protocols occurrences in the connectivity_type property of the same NsVirtualLink node, i.e. the first occurrence of the virtual_link_protocol_data represents the highest layer protocol data, and the last occurrence represents the lowest layer protocol data.</td>
</tr>
</tbody>
</table>

**NOTE 1:** A virtualLinkDescId property, which exists in ETSI GS NFV-IFA 014 [2] is not needed, as the NsVIProfile is contained in the NsVirtualLink node.

**NOTE 2:** The value '1' expresses the highest service availability level.

### 7.2.3.3 Definition

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

```yaml
tosca.datatypes.nfv.NsVIProfile:
  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_level:
      type: integer
      description: Specifies the service availability level for the VL instance created from this profile.
      required: false
  constraints:
    - greater_or_equal: 1

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

### 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>tosca:datatypes.nfv:NsVirtualLinkQos</td>
</tr>
</tbody>
</table>

Table 7.2.5.1-1: Type name, shorthand, and URI

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Specifies the priority level in case of congestion on the underlying physical links.</td>
</tr>
</tbody>
</table>

Table 7.2.5.2-1: Properties

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NsProfile</td>
<td>tosca:nfv:NsProfile</td>
<td>tosca.datatypes.nfv.NsProfile</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_instantiation_level</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>min_number_of_instances</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Minimum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile.</td>
</tr>
<tr>
<td>max_number_of_instances</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Maximum number of instances of the NS based on this NSD that is permitted to exist for this NsProfile.</td>
</tr>
<tr>
<td>flavour_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the applicable network service DF within the scope of the NSD.</td>
</tr>
</tbody>
</table>

7.2.16.3  Definition

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

```yaml
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.
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:Mask</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.Mask</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>starting_point</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>Indicates the offset between the last bit of the source mac address and the first bit of the sequence of bits to be matched.</td>
</tr>
<tr>
<td>length</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>Indicates the number of bits to be matched.</td>
</tr>
<tr>
<td>value</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Provides the sequence of bit values to be matched.</td>
</tr>
</tbody>
</table>

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.
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-v1.3 [20]. 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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsOperationAdditionalParameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsOperationAdditionalParameters</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsOperationAdditionalParameters</td>
</tr>
</tbody>
</table>

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_3
node_types:
tosca.example_NS:
derived_from: tosca.nodes.nfv.NS
properties:

interfaces:
NsLcm:
operations:
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-v1.3 [20].
Table 7.2.19.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsMonitoringParameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsMonitoringParameter</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsMonitoringParameter</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable name of the monitoring parameter.</td>
</tr>
</tbody>
</table>
| performance_metric | yes      | string        | valid values: See YAML definition constraints     | 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:

```yaml
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.

7.2.21 tosca.datatypes.nfv.NsVirtualLinkProtocolData

7.2.21.1 Description
The NsVirtualLinkProtocolData data type describes one protocol layer and associated protocol data for a given virtual link used in a specific NS deployment flavour. Table 7.2.21.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsVirtualLinkProtocolData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.NsVirtualLinkProtocolData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsVirtualLinkProtocolData</td>
</tr>
</tbody>
</table>

7.2.21.2 Properties
The properties of the NsVirtualLinkProtocolData data type shall comply with the provisions set out in table 7.2.21.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>associated_layer_protocol</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Identifies one of the protocols a virtualLink gives access to (ethernet, mpls, odu2, ipv4, ipv6, pseudo-wire) as specified by the connectivity_type property.</td>
</tr>
<tr>
<td>l2_protocol_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.NsL2ProtocolData</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>l3_protocol_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.NsL3ProtocolData</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

7.2.21.3 Definition
The syntax of the NsVirtualLinkProtocolData data type shall comply with the following definition:

```
tosca.datatypes.nfv.NsVirtualLinkProtocolData:
  derived_from: tosca.datatypes.Root
  description: describes one protocol layer and associated protocol data for a given virtual link used in a specific NS 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.NsL2ProtocolData
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.NsL3ProtocolData
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

7.2.21.4 Examples
None.

7.2.21.5 Additional Requirements
None.

7.2.22 tosca.datatypes.nfv.NsL2ProtocolData

7.2.22.1 Description
The NsL2ProtocolData data type describes L2 protocol data for a given virtual link used in a specific NS deployment flavour. Table 7.2.22.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsL2ProtocolData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.NsL2ProtocolData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsL2ProtocolData</td>
</tr>
</tbody>
</table>

7.2.22.2 Properties
The properties of the NsL2ProtocolData data type shall comply with the provisions set out in table 7.2.22.2-1.
Table 7.2.22.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the network name associated with this L2 protocol.</td>
</tr>
<tr>
<td>network_type</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Specifies the network type for this L2 protocol. The value may be overridden at run-time.</td>
</tr>
<tr>
<td>vlan_transparent</td>
<td>no</td>
<td>boolean</td>
<td>default: false</td>
<td>Specifies whether to support VLAN transparency for this L2 protocol or not.</td>
</tr>
<tr>
<td>mtu</td>
<td>no</td>
<td>integer</td>
<td>greater_than: 0</td>
<td>Specifies the Maximum Transmission Unit (MTU) value for this L2 protocol.</td>
</tr>
<tr>
<td>segmentation_id</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

NOTE: If this property is included in the NSD, the property value shall be provided at run-time, unless a default value is provided at design time in the NSD. If a default value is provided at design-time, this value may be overridden at run-time.

7.2.22.3 Definition

The syntax of the NsL2ProtocolData data type shall comply with the following definition:

```yaml
tosca.datatypes.nfv.NsL2ProtocolData:
  derived_from: tosca.datatypes.Root
  description: describes L2 protocol data for a given virtual link used in a specific NS deployment flavour.
  properties:
    name:
      type: string
      description: Identifies the network name associated with this L2 protocol.
      required: false
    network_type:
      type: string
      description: Specifies the network type for this L2 protocol. The value may be overridden at run-time.
      required: false
      constraints:
        - valid_values: [ flat, vlan, vxlan, gre ]
    vlan_transparent:
      type: boolean
      description: Specifies whether to support VLAN transparency for this L2 protocol or not.
      required: false
      default: false
    mtu:
      type: integer
      description: Specifies the maximum transmission unit (MTU) value for this L2 protocol.
      required: false
      constraints:
        - greater_than: 0
    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
```
7.2.22.4 Examples

None.

7.2.22.5 Additional Requirements

None.

7.2.23 `tosca.datatypes.nfv.NsL3ProtocolData`

7.2.23.1 Description

The `NsL3ProtocolData` data type describes L3 protocol data for a given virtual link used in a specific NS deployment flavour. Table 7.2.23.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 7.2.23.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsL3ProtocolData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:NsL3ProtocolData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsL3ProtocolData</td>
</tr>
</tbody>
</table>

7.2.23.2 Properties

The properties of the `NsL3ProtocolData` data type shall comply with the provisions set out in table 7.2.23.2-1.

Table 7.2.23.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifies the network name associated with this L3 protocol.</td>
</tr>
<tr>
<td>ip_version</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>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.</td>
</tr>
<tr>
<td>cidr</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Specifies the CIDR (Classless Inter-Domain Routing) of this L3 protocol. The value may be overridden at run-time.</td>
</tr>
<tr>
<td>ip_allocation_pools</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.NsIpAllocationPool</td>
<td></td>
<td>Specifies the allocation pools with start and end IP addresses for this L3 protocol. The value may be overridden at run-time.</td>
</tr>
</tbody>
</table>

7.2.23.3 Definition

The syntax of the `NsL3ProtocolData` data type shall comply with the following definition:

```yaml
tosca.datatypes.nfv.NsL3ProtocolData:
  derived_from: tosca.datatypes.Root
  description: describes L3 protocol data for a given virtual link used in a specific NS deployment flavour.
  properties:
    name:
      type: string
      description: Identifies the network name associated with this L3 protocol.
      required: false
      ip_version:
```

ETSI
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.NsIpAllocationPool

7.2.23.4 Examples
None.

7.2.23.5 Additional Requirements
None.

7.2.24 tosca.datatypes.nfv.NsIpAllocationPool

7.2.24.1 Description
The NsIpAllocationPool data type specifies a range of IP addresses. Table 7.2.24.1-1 specifies the declared names for
this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 7.2.24.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
</table>

7.2.24.2 Properties
The properties of the NsIpAllocationPool data type shall comply with the provisions set out in table 7.2.24.2-1.

Table 7.2.24.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_ip_address</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>The IP address to be used as the first one in a pool of addresses derived from the cidr block full IP range</td>
</tr>
<tr>
<td>end_ip_address</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>The IP address to be used as the last one in a pool of addresses derived from the cidr block full IP range</td>
</tr>
</tbody>
</table>
7.2.24.3 Definition

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

```python
tosca.datatypes.nfv.NsIpAllocationPool:
    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
```

7.2.24.4 Examples

None.

7.2.24.5 Additional Requirements

None.

7.2.25 tosca.datatypes.nfv.NsScalingAspect

7.2.25.1 Description

The NsScalingAspect data type describes the details of an aspect used for horizontal scaling. Table 7.2.25.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>NsScalingAspect</td>
<td>toscansf:NsScalingAspect</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsScalingAspect</td>
</tr>
</tbody>
</table>

7.2.25.2 Properties

The properties of the NsScalingAspect data type shall comply with the provisions set out in table 7.2.25.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable name of the NS scaling aspect.</td>
</tr>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the NS scaling aspect.</td>
</tr>
<tr>
<td>ns_scale_levels</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.NsLevels</td>
<td></td>
<td>Description of the NS levels for this scaling aspect.</td>
</tr>
</tbody>
</table>
7.2.25.3 Definition

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

```yaml
tosca.datatypes.nfv.NsScalingAspect:
  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
    ns_scale_levels:
      type: map
      description: Description of the NS levels for this scaling aspect.
      required: true
      key_schema:
        type: integer # Integer type in order to number the levels. First level is level 0.
        constraints: # greater_or_equal: 0
      entry_schema:
        type: tosca.datatypes.nfv.NsLevels
```

7.2.25.4 Examples

See clause A.17.

7.2.25.5 Additional Requirements

None.

7.2.26 tosca.datatypes.nfv.NsLevels

7.2.26.1 Description

The NsLevels data type describes the Ns levels. Table 7.2.26.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

Table 7.2.26.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsLevels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.NsLevels</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.NsLevels</td>
</tr>
</tbody>
</table>

7.2.26.2 Properties

The properties of the NsLevels data type shall comply with the provisions set out in table 7.2.26.2-1.

Table 7.2.26.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the NS level.</td>
</tr>
</tbody>
</table>
7.2.26.3 Definition

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

```yaml
tosca.datatypes.nfv.NsLevels:
  derived_from: tosca.datatypes.Root
  description: describes the Ns levels
  properties:
    description:
      type: string
      description: Human readable description of the Ns level
      required: true
```

7.2.26.4 Examples

See clause A.17.

7.2.26.5 Additional Requirements

None.

7.2.27 `tosca.datatypes.nfv.ScaleNsByStepsData`

7.2.27.1 Description

The ScaleNsByStepsData data type describes the information needed to scale an NS instance by one or more scaling steps, with respect to a particular NS scaling aspect as defined in ETSI GS NFV-IFA 013 [i.8]. Table 7.2.27.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaling_direction</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Indicates the type of the scale operation requested.</td>
</tr>
<tr>
<td>aspect</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of the scaling aspect.</td>
</tr>
<tr>
<td>number_of_steps</td>
<td>yes</td>
<td>integer</td>
<td></td>
<td>Number of scaling steps to be executed.</td>
</tr>
</tbody>
</table>

7.2.27.2 Properties

The properties of the ScaleNsByStepsData data type shall comply with the provisions set out in table 7.2.27.2-1.

7.2.27.3 Definition

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

```yaml
tosca.datatypes.nfv.ScaleNsByStepsData:
  derived_from: tosca.datatypes.Root
  description: describes the information needed to scale an NS instance by one or more scaling steps, with respect to a particular NS scaling aspect
```
properties:
  scaling_direction:
    type: string
    description: Indicates the type of the scale operation requested.
    required: true
    constraints:
      - valid_values: [ scale_out, scale_in ]
  aspect:
    type: string
    description: Identifier of the scaling aspect.
    required: true
  number_of_steps:
    type: integer
    description: Number of scaling steps to be executed.
    required: true
    constraints:
      - greater_than: 0
      default: 1

7.2.27.4 Examples
None.

7.2.27.5 Additional Requirements
None.

7.2.28 tosca.datatypes.nfv.ScaleNsToLevelData

7.2.28.1 Description
The ScaleNsByStepsData data type describes the information needed to scale an NS instance to a target size as defined in ETSI GS NFV-IFA 013 [i.8]. Table 7.2.28.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ScaleNsToLevelData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.ScaleNsToLevelData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.ScaleNsToLevelData</td>
</tr>
</tbody>
</table>

7.2.28.2 Properties
The properties of the ScaleNsToLevelData data type shall comply with the provisions set out in table 7.2.28.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instantiation_level</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Identifier of the target instantiation level of the current deployment flavour to which the NS is requested to be scaled. Either instantiation_level or ns_scale_info shall be provided.</td>
</tr>
<tr>
<td>ns_scale_info</td>
<td>no</td>
<td>map of integer</td>
<td></td>
<td>For each scaling aspect of the current deployment flavour, indicates the target scale level to which the NS is to be scaled. Either instantiation_level or ns_scale_info shall be provided.</td>
</tr>
</tbody>
</table>
7.2.28.3 Definition

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

```plaintext
tosca.datatypes.nfv.ScaleNsToLevelData:
  derived_from: tosca.datatypes.Root
  description: describes the information needed to scale an NS instance to a target size.
  properties:
    instantiation_level:
      type: string
      description: Identifier of the target instantiation level of the current deployment flavour to which the NS is requested to be scaled. Either instantiation_level or ns_scale_info shall be provided.
      required: false
    ns_scale_info:
      type: map # key: aspectId
      description: For each scaling aspect of the current deployment flavour, indicates the target scale level to which the NS is to be scaled. Either instantiation_level or ns_scale_info shall be provided.
      required: false
      entry_schema:
        type: integer
        constraints:
          - greater_or_equal: 0
```

7.2.28.4 Examples

None.

7.2.28.5 Additional Requirements

None.

7.3 Artifact Types

None.

7.4 Capability 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-v1.3 [20].
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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Forwarding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.capabilities.nfv.Forwarding</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.Forwarding</td>
</tr>
</tbody>
</table>

#### 7.4.2.2 Properties

None.

#### 7.4.2.3 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>ForwardTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:ForwardTo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.relationships.ForwardTo</td>
</tr>
</tbody>
</table>
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.NsIcm

7.7.1.1 Description

The tosca.interfaces.nfv.NsIcm 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 suffixes 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsIcm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsIcm</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.interfaces.nfv.NsIcm</td>
</tr>
</tbody>
</table>

7.7.1.2 Definition

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

```
tosca.interfaces.nfv.NsIcm:
    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.
```
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.
    # inputs:
    #   additional_parameters:
    #   type: tosca.datatypes.nfv.NsOperationAdditionalParameters
    #   required: false
  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.
    # inputs:
    #   additional_parameters:
    #   type: tosca.datatypes.nfv.NsOperationAdditionalParameters
    #   required: false
  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:
7.7.1.3 Additional Requirements

The implementation and inputs keynames specified in TOSCA-Simple-Profile-YAML-v1.3 [20] 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.

Starting with version 3.3.1 [i.19] of the present document, the Nslcm interface type definition grammar was changed to support notifications and operations. For backward compatibility, TOSCA-Simple-Profile-YAML-v1.3 [20], clause 3.7.5.5 specifies the provisions for handling the previous grammar. Support of the Release 2 Nslcm interface type definition grammar can be removed in subsequent versions of the present document.

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-v1.3 [20]. 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_3

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:
    operations:
      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: ..
 ..

7.7.2 tosca.interfaces.nfv.NsVnfIndicator

7.7.2.1 Description

The tosca.interfaces.nfv.NsVnfIndicator is an empty base interface type for deriving NS specific interface types that include VNF indicator specific notifications which will be used in a NS.

Table 7.7.2.1-1 specifies the declared names for this interface type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NaVnfIndicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsVnfIndicator</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.interfaces.nfv.NsVnfIndicator</td>
</tr>
</tbody>
</table>
7.7.2.2 Definition

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

```tosca
    tosca.interfaces.nfv.NsVnfIndicator:
        derived_from: tosca.interfaces.Root
        description: This interface is an empty base interface type for deriving NS specific interface types that include VNF indicator specific notifications which will be used in a NS.
```

7.7.2.3 Examples

See clause A.15.3.

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:NS</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.NS</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of this NS descriptor. See note 2.</td>
</tr>
<tr>
<td>designer</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the designer of the NSD.</td>
</tr>
<tr>
<td>version</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the version of the NSD.</td>
</tr>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Provides the human readable name of the NSD.</td>
</tr>
<tr>
<td>invariant_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies an NSD in a version independent manner. This attribute is invariant across versions of NSD. See note 2.</td>
</tr>
<tr>
<td>flavour_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of this NS DF within the NSD.</td>
</tr>
<tr>
<td>ns_profile</td>
<td>no</td>
<td>tosca.datatypes.nfv.NsProfile</td>
<td></td>
<td>Specifies a profile of an NS, when this NS is used as nested NS within another NS. See note 1.</td>
</tr>
<tr>
<td>service_availability_level</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 1</td>
<td>If present, specifies the service availability level for the NS instance. See note 3.</td>
</tr>
<tr>
<td>priority</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Specifies the priority for the NS instance. Examples for the usage of priority include conflict resolution in case of resource shortage. See notes 4 and 5.</td>
</tr>
</tbody>
</table>

**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].

**NOTE 3:** The value ‘1’ expresses the highest service availability level.

**NOTE 4:** The value ‘0’ expresses the highest priority and the fact that the NS instance based on this NS DF cannot be pre-empted during resource allocation.

**NOTE 5:** A NSD specific node type definition can further constrain the range of valid values by indicating an upper bound.

### 7.8.1.3 Attributes

The attribute of the NS node type shall comply with the provisions set out in table 7.8.1.3-1.

Table 7.8.1.3-1: Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale_status</td>
<td>no</td>
<td>map of integer</td>
<td></td>
<td>Scale status of the NS, one entry per aspect. Represents for every scaling aspect how “big” the NS has been scaled w.r.t. that aspect.</td>
</tr>
</tbody>
</table>

If the NS supports VNF indicators as the monitoring data as described in ETSI GS NFV-IFA 014 [2], the NS specific node type definition shall include one TOSCA attribute of a primitive type for each monitored VNF indicator.

**NOTE 1:** The value of the attribute for each monitored VNF indicator can be retrieved by passing a path to the get_attribute function: 

get_attribute: [get_attribute: [<VNF_node_template_name>, <attribute_name_defined_in_the_corresponding_VNFD>]]

**NOTE 2:** The rule for naming the attribute for each monitored VNF indicator should be:

"VNF node template name "/"_" attribute name defined in the corresponding VNFD ". See example in clause A.15.3.

**NOTE 3:** 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.
If the scale_status attribute is used in a NS auto-scale policy, the NS specific node type definitions may include additional attribute definitions of type integer, one for each scaling aspect. See example in clause A.15.3.

NOTE 4: As the scale_status attribute is complex, the value of the individual scaling aspects can be retrieved by passing a path to the get_attribute function: [ get_attribute: [ SELF, scale_status, {scaling_aspect} ] ]. If the value of the scale_status property is needed in a constraint (tosca.policies.nfv.NsAutoScale), then its value can be retrieved in an indirect way by accessing the aforementioned additional attributes. This is due to the limitation mentioned in the note 3.

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link</td>
</tr>
</tbody>
</table>

7.8.1.5 Capabilities

None.

7.8.1.6 Definition

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

```yaml
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
```
service_availability_level:
  type: integer
  description: Specifies the service availability level for the NS instance.
  required: false
  constraints:
    - greater_or_equal: 1
priority:
  type: integer
  description: Specifies the priority for the NS instance. Examples for the usage of priority include conflict resolution in case of resource shortage.
  required: false
  constraints:
    - greater_or_equal: 0
attributes:
  scale_status:
    type: map # key: aspectId
    description: Scale status of the NS, one entry per aspect. Represents for every scaling aspect how "big" the NS has been scaled w.r.t. that aspect.
    entry_schema:
      type: integer
      constraints:
        - greater_or_equal: 0
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:

a) The node type shall be derived from: tosca.nodes.nfv NS.

b) 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.

c) Properties listed in in tosca.nodes.nfv NS where the "required:" field is set to "false" may be included.

d) The capabilities, requirements, interfaces of tosca.nodes.nfv NS shall be preserved.

e) 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sap</td>
<td>tosca.nfv:Sap</td>
<td>tosca.nodes.nfv.Sap</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>external_virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability</td>
</tr>
<tr>
<td>internal_virtual_link</td>
<td>yes</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability</td>
</tr>
</tbody>
</table>

7.8.2.5  Capabilities
None.

7.8.2.6  Definition
The syntax of the Sap node type shall comply with the following definition:

```yaml
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 or a corresponding Forwarding node. The first case is illustrated in clause A.7.2 while the second case is illustrated in clause A.14.

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsVirtualLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:NsVirtualLink</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.NsVirtualLink</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vl_profile</td>
<td>yes</td>
<td>tosca.datatype.nfv.NsVlProfile</td>
<td></td>
<td>Specifies instantiation parameters for a virtual link of a particular NS deployment flavour.</td>
</tr>
<tr>
<td>connectivity_type</td>
<td>yes</td>
<td>tosca.datatypes.nfv.ConnectivityType</td>
<td></td>
<td>Network service virtual link connectivity type.</td>
</tr>
<tr>
<td>test_access</td>
<td>no</td>
<td>list of string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Test access facilities available on the VL.</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable information on the purpose of the virtual link (e.g. VL for control plane traffic).</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_linkable</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>VirtualLinkable capability</td>
</tr>
</tbody>
</table>

7.8.3.6 Definition
The syntax of the NsVirtualLink node type shall comply with the following definition:

```yaml
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-v1.3 [20].

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>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NfpPositionElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nodes.nfv.NfpPositionElement</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.NfpPositionElement</td>
</tr>
</tbody>
</table>

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

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile_element</td>
<td>yes</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td></td>
<td>Describes the requirement for the constituent of the NfpPositionElement.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarding</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td></td>
<td>NfpPositionElement forwarding capability</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFP</td>
<td>tosca.nodes.nfv:NFP</td>
<td>tosca.nodes.nfv.NFP</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfp_position</td>
<td>yes</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.8.6.5 Capabilities
None.

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NfpPosition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:NfpPosition</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.NfpPosition</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarding_behaviour</td>
<td>no</td>
<td>string</td>
<td>Possible values: &quot;all&quot;, &quot;lb&quot;, &quot;ff&quot;</td>
<td>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 the referenced CP profile(s). See note.</td>
</tr>
<tr>
<td>forwarding_behaviour _input_parameters</td>
<td>no</td>
<td>map of string</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>yes</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td></td>
<td>Specifies that an NfpPosition requires a node that has a forwarding capability.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarding</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td></td>
<td>NfpPosition forwarding capability</td>
</tr>
</tbody>
</table>

7.8.7.6 Definition

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

```
tosca.nodes.nfv.NfpPosition:
```
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 nested NS node template 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-v1.3 [20].

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>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding</td>
<td>tosca.nodes.nfv:Forwarding</td>
<td>tosca.nodes.nfv:Forwarding</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Capability Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_link</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td>None</td>
<td>Describes the requirement for linking to a virtual linkable node type.</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarding</td>
<td>tosca.capabilities.nfv.Forwarding</td>
<td>None</td>
<td>The forwarding capability exposed by the node.</td>
</tr>
<tr>
<td>virtual_linkable</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td>None</td>
<td>The virtual linkable capability exposed by the node.</td>
</tr>
</tbody>
</table>

7.8.8.6 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsPlacementGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.groups.nfv.NsPlacementGroup</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.groups.nfv.NsPlacementGroup</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the group</td>
</tr>
</tbody>
</table>

7.9.1.3 Definition

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

```yaml
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 either all of the same node type or a combination of tosca.nodes.nfv.VNF and tosca.nodes.nfv.NS node types 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VNFFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VNFFG</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.groups.nfv.VNFFG</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Human readable description of the group</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsAffinityRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:NsAffinityRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsAffinityRule</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsAntiAffinityRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:nfv:NsAntiAffinityRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsAntiAffinityRule</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Yes</td>
<td>String</td>
<td>Valid values : &quot;nfvi_pop&quot;, &quot;zone&quot;, &quot;zone_group&quot;, &quot;nfvi_node&quot;, &quot;network_link_and_node&quot;, &quot;container_namespaces&quot;</td>
<td>Specifies the scope of the local affinity rule. See notes 1 and 2.</td>
</tr>
</tbody>
</table>

NOTE 1: When used in an anti-affinity relationship, the "network_link_and_node" scope is conceptually similar to link and node disjoint paths capabilities used commonly in network Traffic Engineering (TE). For example, as in Fast Reroute Resource Reservation Protocol Traffic Engineering (RSVP-TE) for Label-Switched Path (LSP) tunnels as introduced in IETF RFC 4090 [I.17].

NOTE 2: The "container_namespaces" is only applicable when the targets of the policy are exclusively nodes of type tosca.nodes.nfv.VNF and these are realized exclusively by sets of OS containers. Groups of type tosca.groups.nfv.NsPlacementGroup that consists exclusively of VNFs realized exclusively by set of OS containers may also be target of the policy when the "container_namespaces" scope is indicated.
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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>Yes</td>
<td>string[]</td>
<td>Valid types: tosca.nodes.nfv.VNF, tosca.nodes.nfv.NsVirtualLink, tosca.nodes.nfv.NS, tosca.groups.nfv.NsPlacementGroup</td>
</tr>
</tbody>
</table>

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:

```yaml
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, network_link_and_node, container_namespace ]
        - 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:

```yaml
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, network_link_and_node, container_namespace ]
        - 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:

```yaml
node_templates:
  VNF_1:
    type: tosca.nodes.nfv.exampleVNF

policies:
  policy_affinity_local_VNF_1:
    type: tosca.policies.nfv.NsAffinityRule
    targets: [ VNF_1 ]
    properties:
      scope: nfvi_node
```

The above example illustrates a local affinity rule for all the instances of VNF_1.

```yaml
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.NsPlacementGroup
    members: [ VNF_1, VNF_2 ]

policies:
  policy_antiaffinity_group_1:
    type: tosca.policies.nfv.NsAntiAffinityRule
    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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsSecurityGroupRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.NsSecurityGroupRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsSecurityGroupRule</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string[]</td>
<td>Valid types: tosca.nodes.nfv.Sap.</td>
<td>Target connection points of Sap.</td>
</tr>
</tbody>
</table>

7.10.2.4 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NfpRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.NfpRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NfpRule</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ether_destination_address</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Indicates a destination Mac address.</td>
</tr>
<tr>
<td>ether_source_address</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Indicates a source Mac address.</td>
</tr>
<tr>
<td>ether_type</td>
<td>no</td>
<td>string</td>
<td>ipv4, ipv6</td>
<td>Indicates the protocol carried over the Ethernet layer.</td>
</tr>
<tr>
<td>vlan_tag</td>
<td>no</td>
<td>list of</td>
<td>string</td>
<td>Indicates a VLAN identifier in an IEEE 802.1Q-2014 tag [16]. Multiple tags can be included for QinQ stacking.</td>
</tr>
<tr>
<td>protocol</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Indicates the L4 protocol. For IPv4 [17] this corresponds to the field called &quot;Protocol&quot; to identify the next level protocol. For IPv6 [18] this corresponds to the field called the &quot;Next Header&quot; field. Permitted values: Any keyword defined in the IANA [19] protocol registry, e.g.: • TCP • UDP • ICMP.</td>
</tr>
<tr>
<td>dscp</td>
<td>no</td>
<td>string</td>
<td></td>
<td>For IPv4 [17] a string of &quot;0&quot; and &quot;1&quot; digits that corresponds to the 6-bit Differentiated Services Code. Point (DSCP) field of the IP header. For IPv6 [18] a string of &quot;0&quot; and &quot;1&quot; digits that corresponds to the 6 differentiated services bits of the traffic class header field.</td>
</tr>
<tr>
<td>source_port_range</td>
<td>no</td>
<td>range</td>
<td>0 - 65535</td>
<td>Indicates a range of source ports.</td>
</tr>
<tr>
<td>destination_port_range</td>
<td>no</td>
<td>range</td>
<td>0 - 65535</td>
<td>Indicates a range of destination ports.</td>
</tr>
<tr>
<td>source_ip_address_prefix</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Indicates the source IP address range in CIDR format.</td>
</tr>
<tr>
<td>destination_ip_address_prefix</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Indicates the destination IP address range in CIDR format.</td>
</tr>
<tr>
<td>extended_criteria</td>
<td>no</td>
<td>list of</td>
<td>tosca.data_types.nfv.Mask</td>
<td>Indicates values of specific bits in a frame.</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string[]</td>
<td></td>
<td>The NFPs to which the rule applies.</td>
</tr>
</tbody>
</table>

### 7.10.3.4 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsMonitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsMonitoring</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsMonitoring</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_monitoring_parameters</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.NsMonitoringParameter</td>
<td></td>
<td>Specifies a virtualised resource related performance metric to be monitored on the NS level.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string</td>
<td>Valid types: tosca.nodes.nfv.NS</td>
<td>Specifies the services node type(s) to which the monitoring policy applies.</td>
</tr>
</tbody>
</table>

7.10.4.4 Definition

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

```yaml
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: map #key: id
      description: Specifies a virtualised resource related performance metric to be monitored on the NS level.
      required: true
```
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfMonitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VnfMonitoring</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VnfMonitoring</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnf_monitoring_parameters</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.VnfMonitoringParameter</td>
<td></td>
<td>Specifies a virtualised resource related performance metric to be monitored on the VNF level.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string</td>
<td>Valid types: tosca.nodes.nfv.VNF</td>
<td>Specifies the VNF node type(s) to which the monitoring policy applies.</td>
</tr>
</tbody>
</table>

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
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.

7.10.7 tosca.policies.nfv.NsScalingAspects

7.10.7.1 Description

The NsScalingAspects type is a policy type representing the scaling aspects used for horizontal scaling as defined in ETSI GS NFV-IFA 014 [2]. This policy concerns the whole NS (deployment flavour) represented by the topology_template and thus has no explicit target list. Table 7.10.7.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScalingAspects</td>
<td></td>
</tr>
<tr>
<td>tosca.nfv:NsScalingAspects</td>
<td></td>
</tr>
<tr>
<td>tosca.policies.nfv.NsScalingAspects</td>
<td></td>
</tr>
</tbody>
</table>

7.10.7.2 Properties

The properties of the NsScalingAspects policy type shall comply with the provisions set out in table 7.10.7.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspects</td>
<td>yes</td>
<td>Map of tosca.datatypes.nfv. NsScalingAspect</td>
<td></td>
<td>Describe the details of a particular aspect including the corresponding NS levels.</td>
</tr>
</tbody>
</table>
7.10.7.3 Definition

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

```yaml
tosca.policies.nfv.NsScalingAspects:
  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 014 [2]
  properties:
    aspects:
      type: map # key: aspectId
      description: Describe the details of a particular aspect including the corresponding NS levels.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.NsScalingAspect
      constraints:
        - min_length: 1
```

7.10.7.4 Examples

See clause A.17.

7.10.8 tosca.policies.nfv.VnfToLevelMapping

7.10.8.1 Description

The VnfToLevelMapping type is a policy type representing the number of VNF instances that correspond to every NS level of a particular aspect, as defined in ETSI GS NFV-IFA 014 [2]. Table 7.10.8.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfToLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.VnfToLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VnfToLevelMapping</td>
</tr>
</tbody>
</table>

7.10.8.2 Properties

The properties of the VnfToLevelMapping policy type shall comply with the provisions set out in table 7.10.8.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect:</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Represents the scaling aspect to which this policy applies.</td>
</tr>
<tr>
<td>number_of_instances</td>
<td>yes</td>
<td>map of integer</td>
<td></td>
<td>Describes the number of VNF instances to be deployed for each NS instances of a particular aspect. The first level is level 0.</td>
</tr>
</tbody>
</table>
7.10.8.3 Definition

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

```yaml
tosca.policies.nfv.VnfToLevelMapping:
  derived_from: tosca.policies.Root
  description: The VnfToLevelMapping type is a policy type representing the number of VNF instances to be deployed at each NS level, as defined in ETSI GS NFV-IFA 014 [2]
  properties:
    aspect:
      type: string
      description: Represents the scaling aspect to which this policy applies
      required: true
    number_of_instances:
      type: map # key: Ns level
      description: Number of VNF instances to be deployed for each NS level.
      required: true
      key_schema:
        type: integer # First level is level 0.
        constraints:
          - greater_or_equal: 0
      entry_schema:
        type: integer
        constraints:
          - greater_or_equal: 0
          - min_length: 1
  targets: [ tosca.nodes.nfv.VNF ]
```

7.10.8.4 Examples

See clause A.17.

7.10.9 tosca.policies.nfv.NsToLevelMapping

7.10.9.1 Description

The NsToLevelMapping type is a policy type representing the number of instances of a nested NS that correspond to every NS level of the composite NS of a particular aspect, as defined in ETSI GS NFV-IFA 014 [2]. Table 7.10.9.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsToLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.NsToLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsToLevelMapping</td>
</tr>
</tbody>
</table>

Table 7.10.9.1-1: Type name, shorthand, and URI

7.10.9.2 Properties

The properties of the NsToLevelMapping policy type shall comply with the provisions set out in table 7.10.9.2-1.
Table 7.10.9.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect:</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Represents the scaling aspect to which this policy applies.</td>
</tr>
<tr>
<td>number_ofInstances</td>
<td>yes</td>
<td>map of integer</td>
<td></td>
<td>Describes the number of NS instances of a nested NS to be deployed for each NS level of the composite NS of a particular aspect. The first level is level 0.</td>
</tr>
</tbody>
</table>

7.10.9.3 Definition

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

```yaml
tosca.policies.nfv.NsToLevelMapping:
    derived_from: tosca.policies.Root
    description: The NsToLevelMapping type is a policy type representing the number of NS instances of a nested NS to be deployed at each NS level of the composite NS, as defined in ETSI GS NFV-IFA 014 [2]
    properties:
        aspect:
            type: string
            description: Represents the scaling aspect to which this policy applies
            required: true
        number_of_instances:
            type: map # key: Ns level
            description: Number of NS instances of a nested NS to be deployed for each NS level of the composite NS.
            required: true
            key_schema:
                type: integer # First level is level 0.
                constraints:
                    - greater_or_equal: 0
            entry_schema:
                type: integer
                constraints:
                    - greater_or_equal: 0
                    - min_length: 1
    targets: [ tosca.nodes.nfv.NS ]
```

7.10.9.4 Examples

See clause A.17.

7.10.10 tosca.policies.nfv.VirtualLinkToLevelMapping

7.10.10.1 Description

The VirtualLinkToLevelMapping type is a policy type representing the bitrate requirements of a VL for every NS level of a particular aspect, as defined in ETSI GS NFV-IFA 014 [2]. Table 7.10.10.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

**Table 7.10.10.1-1: Type name, shorthand, and URI**

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinkToLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv: VirtualLinkToLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VirtualLinkToLevelMapping</td>
</tr>
</tbody>
</table>
### 7.10.10.2 Properties

The properties of the VirtualLinkToLevelMapping policy type shall comply with the provisions set out in table 7.10.10.2-1.

#### Table 7.10.10.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect:</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Represents the scaling aspect to which this policy applies.</td>
</tr>
<tr>
<td>bit_rate_requirements</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td></td>
<td>Describes the bitrate requirements of a VL for each NS level of a particular aspect. First level is level 0.</td>
</tr>
</tbody>
</table>

### 7.10.10.3 Definition

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

```yaml
tosca.policies.nfv.VirtualLinkToLevelMapping:
  derived_from: tosca.policies.Root
  description: The VirtualLinkToLevelMapping type is a policy type representing the number of NS instances of a nested NS to be deployed at each NS level of the composite NS, as defined in ETSI GS NFV-IFA 014 [2]
  properties:
    aspect:
      type: string
      description: Represents the scaling aspect to which this policy applies
      required: true
    bit_rate_requirements:
      type: map # key: Ns level
      description: Bitrate requirements of a VL for each NS level.
      required: true
      key_schema:
        type: integer # First level is level 0.
      entry_schema:
        type: tosca.datatypes.nfv.LinkBitrateRequirements
        constraints:
          - min_length: 1
    targets: [ tosca.nodes.nfv.NsVirtualLink ]
```

### 7.10.10.4 Examples

See clause A.17.

### 7.10.11 tosca.policies.nfv.NsInstantiationLevels

#### 7.10.11.1 Description

The NsInstantiationLevels 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 VNF instances and nested NS instances to be created as defined in ETSI GS NFV-IFA 014 [2]. This policy concerns the whole NS(deployment flavour) represented by the topology_template and thus has no explicit target list. Table 7.10.11.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].
Table 7.10.11.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsInstantiationLevels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:NsInstantiationLevels</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsInstantiationLevels</td>
</tr>
</tbody>
</table>

7.10.11.2 Properties

The properties of the NsInstantiationLevels policy type shall comply with the provisions set out in table 7.10.11.2-1.

Table 7.10.11.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns_levels</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.NsLevels</td>
<td></td>
<td>Describes the various levels of resources that can be used to instantiate the NS using this flavour.</td>
</tr>
<tr>
<td>default_level</td>
<td>no</td>
<td>string</td>
<td></td>
<td>The default instantiation level for this flavour.</td>
</tr>
</tbody>
</table>

7.10.11.3 Definition

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

```yaml
tosca.policies.nfv.NsInstantiationLevels:
  derived_from: tosca.policies.Root
  description: The NsInstantiationLevels 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 VNF and nested NS instances to be created as defined in ETSI GS NFV-IFA 014 [2].
  properties:
    ns_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.NsLevels
      constraints:
        - min_length: 1
    default_level:
      type: string # levelId
      description: The default instantiation level for this flavour.
      required: false # required if multiple entries in ns_levels
```

7.10.12 tosca.policies.nfv.VnfToInstantiationLevelMapping

7.10.12.1 Description

The VnfToInstantiationLevelMapping type is a policy type representing the number of VNF instances that correspond to every NS instantiation level, as defined in ETSI GS NFV-IFA 014 [2]. Table 7.10.12.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].
Table 7.10.12.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfToInstantiationLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.VnfToInstantiationLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VnfToInstantiationLevelMapping</td>
</tr>
</tbody>
</table>

7.10.12.2 Properties

The properties of the VnfToInstantiationLevelMapping policy type shall comply with the provisions set out in table 7.10.12.2-1.

Table 7.10.12.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
</table>
| number_of_instances   | yes      | map of integer  | - greater_or_equal: 0  
- min_length: 1          | Describes the number of VNF instances to be deployed for each NS instantiation level. |

7.10.12.3 Definition

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

```yaml
tosca.policies.nfv.VnfToInstantiationLevelMapping:
  derived_from: tosca.policies.Root
  description: The VnfToInstantiationLevelMapping type is a policy type representing the number of VNF instances to be deployed at each NS instantiation level, as defined in ETSI GS NFV-IFA 014 [2]
  properties:
    number_of_instances:
      type: map # key: Ns instantiation level
      description: Number of VNF instances to be deployed for each NS instantiation level.
      required: true
      entry_schema:
        type: integer
        constraints:
        - greater_or_equal: 0
        - min_length: 1
    targets: [ tosca.nodes.nfv.VNF ]
```

7.10.12.4 Examples

See clause A.17.

7.10.12.5 Additional requirements

There shall be one VnfToInstantiationLevelMapping policy defined for each VNF of the NS. If no instances of a given VNF have to be deployed at NS instantiation time for a certain instantiation level, the number_of_instances in the corresponding map entry shall be set to 0.

7.10.13 tosca.policies.nfv.NsToInstantiationLevelMapping

7.10.13.1 Description

The NsToInstantiationLevelMapping type is a policy type representing the number of instances of a nested NS that correspond to every NS instantiation level of the composite NS, as defined in ETSI GS NFV-IFA 014 [2].
Table 7.10.13.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsToInstantiationLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.NsToInstantiationLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsToInstantiationLevelMapping</td>
</tr>
</tbody>
</table>

### 7.10.13.2 Properties

The properties of the NsToInstantiationLevelMapping policy type shall comply with the provisions set out in table 7.10.13.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_of_instances</td>
<td>yes</td>
<td>map of integer</td>
<td></td>
<td>Describes the number of NS instances of a nested NS to be deployed for each NS instantiation level of the composite NS.</td>
</tr>
</tbody>
</table>

### 7.10.13.3 Definition

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

```yaml
tosca.policies.nfv.NsToInstantiationLevelMapping:
  derived_from: tosca.policies.Root
  description: The NsToInstantiationLevelMapping type is a policy type representing the number of NS instances of a nested NS to be deployed at each NS instantiation level of the composite NS, as defined in ETSI GS NFV-IFA 014 [2]
  properties:
    number_of_instances:
      type: map # key: Ns instantiation level
      description: Number of NS instances of a nested NS to be deployed for each NS instantiation level of the composite NS.
      required: true
      entry_schema:
        type: integer
        constraints:
          - greater_or_equal: 0
          - min_length: 1
        targets: [ tosca.nodes.nfv.NS ]
```

### 7.10.13.4 Examples

See clause A.17.

### 7.10.13.5 Additional requirements

There shall be one NsToInstantiationLevelMapping policy defined for each nested NS of the composite NS. If no instances of a given nested NS have to be deployed at NS instantiation time for a certain instantiation level, the number_of_instances in the corresponding map entry shall be set to 0.
7.10.14  

**tosca.policies.nfv.VirtualLinkToInstantiationLevelMapping**

7.10.14.1 Description

The VirtualLinkToInstantiationLevelMapping type is a policy type representing the bitrate requirements of a VL for every NS instantiation level of a particular aspect, as defined in ETSI GS NFV-IFA 014 [2]. Table 7.10.14.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinkToInstantiationLevelMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv: VirtualLinkToInstantiationLevelMapping</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.VirtualLinkToInstantiationLevelMapping</td>
</tr>
</tbody>
</table>

7.10.14.2 Properties

The properties of the VirtualLinkToInstantiationLevelMapping policy type shall comply with the provisions set out in table 7.10.14.2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit_rate_requirements</td>
<td>yes</td>
<td>map of tosca.datatypes.nfv.LinkBitrateRequirements</td>
<td>Describes the bitrate requirements of a VL for each NS instantiation level.</td>
<td></td>
</tr>
</tbody>
</table>

7.10.14.3 Definition

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

```yaml
tosca.policies.nfv.VirtualLinkToInstantiationLevelMapping:
  derived_from: tosca.policies.Root
  description: The VirtualLinkToInstantiationLevelMapping type is a policy type describing the bitrate requirements of a VL at each NS instantiation level of the composite NS, as defined in ETSI GS NFV-IFA 014 [2]
  properties:
    bit_rate_requirements:
      type: map # key: Ns instantiation level
      description: Bitrate requirements of a VL for each NS instantiation level.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.LinkBitrateRequirements
      constraints:
        - min_length: 1
    targets: [ tosca.nodes.nfv.NsVirtualLink ]
```

7.10.14.4 Examples

See clause A.17.
7.10.15 tosca.policies.nfv.NsAutoScale

7.10.15.1 Description

The NsAutoScale is a base policy type for defining NS auto-scale specific policies as defined in ETSI GS NFV-IFA 014 [2].

Table 7.10.15.1-1 specifies the declared names for this policy type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>NsAutoScale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.NsAutoScale</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.NsAutoScale</td>
</tr>
</tbody>
</table>

7.10.15.2 Properties

None.

7.10.15.3 Definition

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

```yaml
tosca.policies.nfv.NsAutoScale:
  derived_from: tosca.policies.Root
  description: The NsAutoScale policy type is a base policy type for defining NS auto-scale specific policies as defined in ETSI GS NFV-IFA 014 [2].
  targets: [ tosca.nodes.nfv.NS ]
```

7.10.15.4 Additional requirements

The NSD service template may include specific policies of NsAutoScale type with the following requirements:

a) it shall include one or more trigger definitions which:

- shall include an event with a value equal to the full name of a notification in the NsVnfIndicator interface definition of the NS node where the policy applies;
- may include a condition definition which can assert the value of vnf indicator attributes and other node attributes using arbitrary AND and OR combinations of the individual assertions;
- may include an action invoking one or multiple operations of the NsLcm interface;

b) the target shall be set to the node template to which the policy applies, i.e. to the node template of the NS specific type present in the topology template that represents a particular deployment flavour.

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 two levels of
service templates

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. The file names of all the lower
service templates shall be declared as the value of the Other-Definitions key in the TOSCA.meta file of the NSD file
structure as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

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-v1.3 [20] and shall include:

- an import statement referencing the TOSCA types definition file as defined in clause B.3;
- an import statement referencing a yaml file which contains an NS specific node type definition;
- a topology template with a node template of the NS-specific node type, which
  - shall include the flavour_id and other properties that are marked as required but do not have a default
    value in the VNF specific node type definition;
  - shall include the requirements as defined in clause 7.8.1;
  - may include other properties specified in the NS specific node type definition, excluding the
    'service_availability_level', 'ns_profile' and the 'priority' properties; and
  - may include a substitute directive;
- optionally, additional NS-specific type definitions and import statements referencing additional NS-specific
  files containing only type definitions used by this TOSCA service template;
- optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3
  [20].

Irrespective of the presence of absence of the substitute directive, 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 as declared
in the Other-Definitions of the TOSCA.meta file in the NSD file structure. The NSD consumer shall silently ignore the
substitute directive if explicit directives are not supported.

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-v1.3 [20] and shall include:

- an import statement referencing the TOSCA types definition file as defined in clause B.3;
- 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;
- 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. To facilitate parsing the NSD, the imported files may be included in the NSD file structure as
specified in ETSI GS NFV-SOL 007 [i.11], in which case the node type definitions they contain shall be
identical to those contained in the corresponding VNF package, PNF archive and NSD file structure.
NOTE 1: These node type definitions are only used to parse the NSD. For the LCM operations of the corresponding VNF or nested NS, such node type definitions do not apply, only the files included in the related VNF package or NSD file structure will be used.

d) optionally, additional NS-specific type definitions and import statements referencing additional NS-specific files containing type definitions used by this TOSCA service template; and

e) 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 and its value as defined in substitution_filter which identifies the DF corresponding to this low level template within the NSD;

NOTE 2: Starting with version 3.3.1 of the present document, the property_mapping grammar was changed to support substitution_filter. For backward compatibility, TOSCA-Simple-Profile-YAML-v1.3 [20], clause 3.8.8.3. specifies the provisions for handling the previous grammar. Support of the Release 2 property_mapping grammar can be removed in subsequent versions of the present document.
   - the mapping of the virtual_link requirements on SAPs;

NOTE 3: When a Sap re-exposes a VNF or PNF external connection point or a Sap of a nested NS, the virtual_link requirement of the NS is mapped to the virtual_link requirement of the VNF or PNF or nested NS node or a corresponding Forwarding node (i.e. the forwarding node that has a capability matching the virtual link requirement of the node whose external connection point or Sap is been exposed). Otherwise the virtual_link requirement of the NS is mapped to the external_virtual_link requirement of the SAP node.
   - a node template referencing the NS specific node type containing the 'service_availability_level' property with a value, if applicable; the 'priority' property with a value, if applicable; implementations of the operations of the LCM interface to be executed by the NFVO, if applicable;
   - 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-v1.3 [20] 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; and
   - additional group definitions, policy definitions and parameter definitions if applicable.

f) optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

Either the service_availability_level property in the NS node template or the service_availability_level attribute in any of the VNF node templates may be provided, but not both. Either the service_availability_level property in the NS node template or the service_availability_level attribute in any of the NsVirtualLink node templates may be provided, but not both.

NOTE 4: The format and structure of an NSD file structure is defined in ETSI GS NFV-SOL 007 [i.11].

NOTE 5: 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 filter for selecting a particular lower level TOSCA service template inside the NSD file structure as described in TOSCA-Simple-Profile-YAML-v1.3 [20].

### 7.11.3 Single deployment flavour design with one service template

In case of the single deployment flavour scenario with one service template design, the NSD shall use TOSCA-Simple-Profile-YAML-v1.3 [20] and shall include:

a) an import statement referencing the TOSCA types definition file as defined in clause B.3:
b) 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. To facilitate parsing the NSD, the imported files may be included in the NSD file structure as specified in ETSI GS NFV-SOL 007 [i.11], in which case the node type definitions they contain shall be identical to those contained in the corresponding VNF package, PNF archive and NSD file structure.

NOTE 1: These node type definitions are only used to parse the NSD. For the LCM operations of the corresponding VNF or nested NS, such node type definitions do not apply, only the files included in the related VNF package or NSD file structure will be used.

c) optionally, additional VNF-specific type definitions and import statements referencing additional NS-specific files containing only type definitions used by this TOSCA service template;

d) an NS specific node type definition derived from the tosca.nodes.nfv.NS node type, as defined in clause 7.8.1; and

e) 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;

NOTE 2: When a Sap re-exposes a VNF or PNF external connection point or a Sap of a nested NS, the virtual_link requirement of the NS is mapped to the virtual_link requirement of the VNF or PNF or nested NS node or a corresponding Forwarding node (i.e. the forwarding node that has a capability matching the virtual link requirement of the node whose external connection point or Sap is been exposed). Otherwise the virtual_link requirement of the NS is mapped to the external_virtual_link requirement of the SAP node.
   - 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-v1.3 [20] 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;
   - additional group definitions, policy definitions and parameter definitions if applicable;

f) optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

See clause A.8 for an example of NSD design with single deployment flavour.

NOTE 3: The service template is deployed stand-alone, i.e. without performing a substitution. However including the substitution_mappings rule indicates its ability to substitute a node template of the NS specific node type, which may appear in an NSD.

NOTE 4: 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-Profile-YAML-v1.3 [20] to the PNFD.
Table 8.1-1: Mapping of ETSI GS NFV-IFA 014 [2] information elements with TOSCA types

<table>
<thead>
<tr>
<th>ETSI NFV Information Element ETSI GS NFV-IFA 014 [2]</th>
<th>TOSCA type</th>
<th>Derived from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pnf</td>
<td>tosca.nodes.nfv.PNF</td>
<td>tosca.nodes.Root</td>
</tr>
<tr>
<td>PnfExtCpd (PNF External Connection Point)</td>
<td>tosca.nodes.nfv.PnfExtCp</td>
<td>tosca.nodes.Root</td>
</tr>
</tbody>
</table>

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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocationInfo</td>
<td>tosca.datatypes.nfv.LocationInfo</td>
<td>tosca.datatypes.nfv.LocationInfo</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>country_code</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Shall be a two-letter ISO 3166 [10] country code in capital letters.</td>
</tr>
<tr>
<td>civic_address_element</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.CivicAddressElement</td>
<td>Elements composing the civic address where the PNF is deployed.</td>
<td></td>
</tr>
<tr>
<td>geographic_coordinates</td>
<td>no</td>
<td>tosca.datatypes.nfv.GeographicCoordinates</td>
<td>Geographic coordinates (e.g. Altitude, Longitude, Latitude) where the PNF is deployed.</td>
<td></td>
</tr>
</tbody>
</table>

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
8.2.5.4 Examples

```yaml
<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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>CivicAddressElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv: CivicAddressElement</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.CivicAddressElement</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca_type</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Describe the content type of caValue. The value of caType shall comply with section 3.4 of IETF RFC 4776 [11].</td>
</tr>
<tr>
<td>ca_value</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Content of civic address element corresponding to the caType. The format caValue shall comply with section 3.4 of IETF RFC 4776 [11].</td>
</tr>
</tbody>
</table>

8.2.6.3 Definition

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

```yaml
tosca.datatypes.nfv.CivicAddressElement:
  derived_from: tosca.datatypes.Root
  properties:
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>GeographicCoordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv.GeographicCoordinates</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.GeographicCoordinates</td>
</tr>
</tbody>
</table>

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.

| Name                   | Required | Type       | Constraints                                                                 | Description                                                                 |}
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude_uncertainty</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Describe the content of latitude_uncertainty. The value of latitude_uncertainty shall comply with LatUnc in section 2.3 of IETF RFC 6225 [21].</td>
</tr>
<tr>
<td>latitude</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Describe the content of latitude. The value of latitude shall comply with Latitude in section 2.3 of IETF RFC 6225 [21].</td>
</tr>
<tr>
<td>longitude_uncertainty</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Describe the content of longitude_uncertainty. The value of longitude_uncertainty shall comply with LongUnc in section 2.3 of IETF RFC 6225 [21].</td>
</tr>
<tr>
<td>longitude</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Describe the content type of longitude. The value of longitude shall comply with Longitude in section 2.3 of IETF RFC 6225 [21].</td>
</tr>
<tr>
<td>altitude_type</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>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].</td>
</tr>
<tr>
<td>altitude_uncertainty</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Describe the content of altitude_uncertainty. The value of altitude_uncertainty shall comply with AltUnc in section 2.4 of IETF RFC 6225 [21].</td>
</tr>
<tr>
<td>altitude</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Describe the content of altitude. The value of altitude shall comply with Altitude in section 2.4 of IETF RFC 6225 [21].</td>
</tr>
</tbody>
</table>
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 Capability 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>PNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca:PNF</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.PNF</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descriptor_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of this PNFD information element. It uniquely identifies the PNFD.</td>
</tr>
<tr>
<td>function_description</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Describes the PNF function.</td>
</tr>
<tr>
<td>provider</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the provider of the PNFD.</td>
</tr>
<tr>
<td>version</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifies the version of the PNFD.</td>
</tr>
<tr>
<td>descriptor_invariant_id</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Identifier of this PNFD in a version independent manner. This attribute is invariant across versions of PNFD. See note.</td>
</tr>
<tr>
<td>name</td>
<td>yes</td>
<td>string</td>
<td></td>
<td>Name to identify the PNFD.</td>
</tr>
<tr>
<td>geographical_location_info</td>
<td>no</td>
<td>tosca.datatype.nfv.LocationInfo</td>
<td></td>
<td>Provides information about the geographical location (e.g. geographic coordinates or address of the building, etc.) of the PNF.</td>
</tr>
</tbody>
</table>

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 [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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Describes the requirements for linking to virtual link</td>
</tr>
</tbody>
</table>

8.8.1.5 Capabilities

None.

8.8.1.6 Definition

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

```plaintext
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:
```

Ref. [9]
8.8.1.7 Artifact

None.

8.8.1.8 Additional Requirements

For a given PNFD, a new PNF node type shall be defined following the below requirements:

a) The node type shall be derived from: tosca.nodes.nfv.PNF.

b) All properties listed in tosca.nodes.nfv.PNF where the "required:" field is set to "true" shall be included with their values indicates as constraints.

c) The requirements of tosca.nodes.nfv.PNF shall be preserved.

d) Depending on the number of external connection points of the PNF that need to connect to NS virtual links, additional requirements for VirtualLinkable capability shall be defined. In this case, it is the 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-v1.3 [20].
### Table 8.8.2.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>PnfExtCp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanyf:PnfExtCp</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.nodes.nfv.PnfExtCp</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Capability type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>external_virtual_link</td>
<td>no</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
<td></td>
<td>Specifies that CP instances require to be connected to a node that has a VirtualLinkable capability.</td>
</tr>
</tbody>
</table>

#### 8.8.2.5 Capabilities

None.

#### 8.8.2.6 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>PnfSecurityGroupRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.PnfSecurityGroupRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.PnfSecurityGroupRule</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targets</td>
<td>yes</td>
<td>string</td>
<td>Valid types: tosca.nodes.nfv.PnfExtCp</td>
<td>Target connection points of PnfExtCp</td>
</tr>
</tbody>
</table>

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-v1.3 [20] and includes:

a)   an import statement referencing the TOSCA types definition file as defined in clause B.4;

b)   a PNF specific node type definition derived from the tosca.nodes.nfv.PNF node type, as defined in clause 8.8.1; and

c)   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;

d)   optionally, metadata and dsl definitions as specified in section 3.10 of TOSCA-Simple-Profile-YAML-v1.3 [20].

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

NOTE:   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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>L2AddressData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.L2AddressData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.L2AddressData</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mac_address_assignment</td>
<td>yes</td>
<td>boolean</td>
<td>-</td>
<td>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.</td>
</tr>
</tbody>
</table>

9.2.1.3 Definition

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

```yaml
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

```yaml
<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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>L3AddressData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.L3AddressData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.L3AddressData</td>
</tr>
</tbody>
</table>
### 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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip_address_assignment</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>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. See note.</td>
</tr>
<tr>
<td>floating_ip_activated</td>
<td>yes</td>
<td>boolean</td>
<td></td>
<td>Specify if the floating IP scheme is activated on the Connection Point or not. See note.</td>
</tr>
<tr>
<td>ip_address_type</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Define address type. The address type should be aligned with the address type supported by the layer_protocols properties of the connection point. See note.</td>
</tr>
<tr>
<td>number_of_ip_address</td>
<td>no</td>
<td>Integer</td>
<td>greater_than: 0</td>
<td>Minimum number of IP addresses to be assigned. See note.</td>
</tr>
<tr>
<td>fixed_ip_address</td>
<td>no</td>
<td>list of string</td>
<td></td>
<td>IP address to be assigned to the CP instance. This attribute is only permitted for Cpds without external connectivity, i.e. connectivity outside the VNF. If present, it shall be compatible with the values of the L3ProtocolData of the VnfVirtualLink referred to by the Cp, if L3ProtocolData is included in the VnfVirtualLink. See note.</td>
</tr>
</tbody>
</table>

**NOTE:** If the fixed_ip_address attribute is present:
- the ip_address_assignment attribute shall be set to True, as the value assignment is still handled by the VNFM based on the fixed_ip_address attribute value;
- the value of the floating_ip_activated attribute shall be set to false;
- the value of the ip_address_type attribute, if present, shall be set consistently with the fixed_ip_address;
- the value of the number_of_ip_address attribute, if present, shall be set consistently with the cardinality of the fixed_ip_address.

### 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
```
9.2.2.4 Examples

The following is an example of the case using dynamic IP address.

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

The following is an example of the case using fixed IP address.

```yaml
<some_tosca_entity>:
  properties:
    l3_address_data:
      ip_address_assignment: true
      floating_ip_activated: false
      ip_address_type: ipv4
      number_of_ip_address: 1
      fixed_ip_address:
        - 192.168.0.1
```

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-v1.3 [20].
Table 9.2.3.1-1: Type name, shorthand, and URI

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>AddressData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:AddressData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.AddressData</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address_type</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>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.</td>
</tr>
<tr>
<td>l2_address_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.L2AddressData</td>
<td>Shall be present when the address_type is mac_address.</td>
<td>Provides the information on the MAC addresses to be assigned to a connection point.</td>
</tr>
<tr>
<td>l3_address_data</td>
<td>no</td>
<td>tosca.datatypes.nfv.L3AddressData</td>
<td>Shall be present when the address_type is ip_address.</td>
<td>Provides the information on the IP addresses to be assigned to a connection point.</td>
</tr>
</tbody>
</table>

9.2.3.3 Definition

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

```yaml
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

The following is an example of the case using dynamic IP address.
The following is an example of the case using fixed IP address.

```yaml
<some_tosca_entity>:
  properties:
    address_data:
      address_type: ip_address
    l3_address_data:
      ip_address_assignment: true
      floating_ip_activated: false
      ip_address_type: ipv4
      number_of_ip_address: 1
      fixed_ip_address:
        - 192.168.0.1
```

### 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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectivityType</td>
<td>tosca.datatypes.nfv.ConnectivityType</td>
<td>tosca.datatypes.nfv.ConnectivityType</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layer_protocols</td>
<td>yes</td>
<td>list of string</td>
<td>Valid values: See YAML definition</td>
<td>Identifies the protocol a virtualLink gives access to (ethernet, mpls, odu2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>constraints</td>
<td>ipv4, ipv6, pseudo-wire). The top layer protocol of the virtualLink protocol</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>stack shall always be provided. The lower layer protocols may be included</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when there are specific requirements on these layers. See note.</td>
</tr>
<tr>
<td>flow_pattern</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition</td>
<td>Identifies the flow pattern of the connectivity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>constraints</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: If more than 1 values are present, the first value represents the highest layer protocol data, and the last value represents the lowest layer protocol data.

9.2.4.3 Definition

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

```yaml
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

```yaml
<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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>LinkBitrateRequirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:LinkBitrateRequirements</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.LinkBitrateRequirements</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>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).</td>
</tr>
<tr>
<td>leaf</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>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).</td>
</tr>
</tbody>
</table>

9.2.5.3  Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>CpProtocolData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:CpProtocolData</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.CpProtocolData</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>associated_layer_protocol</td>
<td>yes</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>One of the values of the property layer_protocols of the CP.</td>
</tr>
<tr>
<td>address_data</td>
<td>no</td>
<td>list of tosca.datatypes.nfv.AddressData</td>
<td></td>
<td>Provides information on the addresses to be assigned to the CP.</td>
</tr>
</tbody>
</table>

9.2.6.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Qos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>toscanfv:Qos</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.Qos</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>latency</td>
<td>yes</td>
<td>scalar-unit.time</td>
<td>greater_than: 0 s</td>
<td>Specifies the maximum latency.</td>
</tr>
<tr>
<td>packet_delay_variation</td>
<td>yes</td>
<td>scalar-unit.time</td>
<td>greater_or_equal: 0 s</td>
<td>Specifies the maximum jitter.</td>
</tr>
<tr>
<td>packet_loss_ratio</td>
<td>no</td>
<td>float</td>
<td>in_range: [0.0, 1.0]</td>
<td>Specifies the maximum packet loss ratio.</td>
</tr>
</tbody>
</table>

9.2.7.3 Definition

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

```yaml
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-v1.3 [20].

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

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnfProfile</td>
<td>tosca.datatypes.nfv.VnfProfile</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instantiation_</td>
<td>no</td>
<td>string</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min_number_of_</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Minimum number of instances of the VNF based on this VNFD that is permitted to exist for this VnfProfile.</td>
</tr>
<tr>
<td>instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_number_of_</td>
<td>yes</td>
<td>integer</td>
<td>greater_or_equal: 0</td>
<td>Maximum number of instances of the VNF based on this VNFD that is permitted to exist for this VnfProfile.</td>
</tr>
<tr>
<td>instances</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service_availability_level</td>
<td>no</td>
<td>integer</td>
<td>greater_or_equal: 1</td>
<td>If present, specifies the service availability level for the VNF instance created from this profile. See note.</td>
</tr>
</tbody>
</table>

NOTE: The value ‘1’ expresses the highest service availability level.

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

```yaml
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
```

ETSIs
### 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VnfMonitoringParameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.datatypes.nfv.VnfMonitoringParameter</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.datatypes.nfv.VnfMonitoringParameter</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable name of the monitoring parameter.</td>
</tr>
<tr>
<td>performance_metric</td>
<td>yes</td>
<td>string</td>
<td>valid values: See YAML definition constraints</td>
<td>Identifies a performance metric to be monitored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>collection_period</td>
<td>no</td>
<td>scalar-unit.time</td>
<td></td>
<td>Describes the periodicity at which to collect the performance information.</td>
</tr>
</tbody>
</table>

9.2.9.3 Definition

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

```python
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 Capability 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinkable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VirtualLinkable</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.capabilities.nfv.VirtualLinkable</td>
</tr>
</tbody>
</table>

9.4.1.2 Properties

None.

9.4.1.3 Definition

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

```yaml
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 or the VnfExtCp and the VnfVirtualLink node types or the association between either the VnfExtCp, the VduCp, the VirtualCp, the PnfExtCp or the 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VirtualLinksTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VirtualLinksTo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.relationships.nfv.VirtualLinksTo</td>
</tr>
</tbody>
</table>

9.6.1.2 Properties

None.

9.6.1.3 Definition

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

```yaml
tosca.relationships.nfv.VirtualLinksTo:
  derived_from: tosca.relationships.DependsOn
  description: Represents an association relationship between the VduCp or the VnfExtCp and the VnfVirtualLink node types or the association between either a VnfExtCp, a VduCp, a VirtualCp, 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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>VipVirtualLinksTo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.nfv:VipVirtualLinksTo</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.relationships.nfv.VipVirtualLinksTo</td>
</tr>
</tbody>
</table>

9.6.3.2 Properties

None.
9.6.3.3 Definition

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

```yaml
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Type Qualified Name</th>
<th>Type URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cp</td>
<td>tosca.nfv.Cp</td>
<td>tosca.nodes.nfv.Cp</td>
</tr>
</tbody>
</table>

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

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>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>layer_protocols</td>
<td>yes</td>
<td>list of string</td>
<td>Valid values: See YAML definition constraints</td>
<td>Identifies which protocol the connection point uses for connectivity purposes.</td>
</tr>
<tr>
<td>role</td>
<td>no</td>
<td>string</td>
<td>Valid values: See YAML definition constraints</td>
<td>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.</td>
</tr>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Provides human-readable information on the purpose of the connection point (e.g. connection point for control plane traffic).</td>
</tr>
</tbody>
</table>
### 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:

```yaml
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
```
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-v1.3 [20].

<table>
<thead>
<tr>
<th>Shorthand Name</th>
<th>Abstract.SecurityGroupRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Qualified Name</td>
<td>tosca.policies.nfv.Abstract.SecurityGroupRule</td>
</tr>
<tr>
<td>Type URI</td>
<td>tosca.policies.nfv.Abstract.SecurityGroupRule</td>
</tr>
</tbody>
</table>

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.

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
Table 9.10.1.2-1: Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Constraints</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>no</td>
<td>string</td>
<td></td>
<td>Human readable description of the security group rule.</td>
</tr>
<tr>
<td>direction</td>
<td>yes</td>
<td>string</td>
<td>ingress, egress</td>
<td>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.</td>
</tr>
<tr>
<td>ether_type</td>
<td>yes</td>
<td>string</td>
<td>ipv4, ipv6</td>
<td>Indicates the protocol carried over the Ethernet layer.</td>
</tr>
<tr>
<td>protocol</td>
<td>yes</td>
<td>string</td>
<td>see note</td>
<td>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.</td>
</tr>
<tr>
<td>port_range_min</td>
<td>yes</td>
<td>integer</td>
<td>0 - 65535</td>
<td>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.</td>
</tr>
<tr>
<td>port_range_max</td>
<td>yes</td>
<td>integer</td>
<td>0 - 65535</td>
<td>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.</td>
</tr>
</tbody>
</table>

NOTE: "protocol" constraints values: hopopt, icmp, igmp, ggp, ipv4, st, tcp, cbt, egp, igp, bbn_rcc_mon, nvp_i, pup, argus, emcon, xnet, chaos, udp, mux, dcn_meas, hmp, prm, xns_idp, trunk_1, trunk_2, leaf_1, leaf_2, rdp, iftp, iso_tp4, netbit, mfe_nsp, merit_inp, dcpp, 3pc, idpr, xtp, ddp, idpr_cntp, tp++, il, ipv6, sdrp, ipv6_route, ipv6_frag, idrp, rsvp, gre, dsr, bna, esp, ah, i, n lsp, swipe, nar p, mobile, tsp, sdp, ipv6_icmp, ipv6_no_nxt, ipv6_opts, cftp, sat_expak, kryptolan, rvd, ippc, sat_mon, visa, ipv6, cpmx, cphb, wsn, pvp, br_sat_mon, sun_nd, wb_mon, wb_expak, iso_ip, vtmtp, secure_vmtmp, vimes, tpt, iptm, nsfnet_igp, dgp, tcf, eigrp, ospf, ospf_def, sprite_rpc, larp, mtp, ax, i25, ipv6, micp, scv, sp, etherip, encap, gmp, vfmp, fnmp, fnmr, pm, aris, scps, oox, a/h, ip, cmpx, snp, vmpq, pe, ipv6_in_ipv6, vrrp, pgm, l2tp, ddx, iatp, stp, sdp, uts, sm, ptp, isis, fire, cftp, mtdp, sscompc, ipt, sdp, pipe, scftp, fc, rsvp, e2e_ignore, mobility, udp_lite, mpls_in_ipv6, manet, hip, shim6, weep, rohc.

9.10.1.3 Definition

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

```yaml
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.
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, rdq, irtp, iso_tp4, netb1t, mfe_nsp, merit_1np, dscp, 3pc, idpr, xtp, ddp, idpr_cmp, tp++, il, ipv6, sdrp, ipv6_route, ipv6_frag, idrp, rsvp, gre, dsr, bna, esp, ah, i_nsp, swipe, narp, mobile, tsp, skip, ipv6_icmp, ipv6_no NXT, ipv6_opts, cftp, sat_expak, kryptolan, rvd, ippc, sat_mon, visa, ipcv, crpv, crph, wsn, pvp, br_sat_mon, sun nd, wb_mon, wb_expak, iso_ip, vmtt, secure_vmtt, vines, ttp, iptm, nsfnet_igp, dgp, tcf, eigrp, ospf, bgp, sprite_rpc, lar, mtp, ax.25, ipip, miip, scc_sp, etherip, encap, gmp, ifmp, pmni, pm, aris, scps, qnx, a/n, ip_comp, snp, compaq_peer, ipx in ip, vrrp, pgm, l2tp, ddx, iatp, stp, sdp, uti, smp, sm, ppt, isis, fire, crtp, crudp, sscp, smp, ipt, sps, pipe, scp, fc, rsvp_e2e Ignore, mobility, udp_lite, mpls in ip, manet, hip, shim6, wesip, 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/NS deployment flavour elements to TOSCA concept.

A.1.2 Design principle for VNF 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 scalingAspects, 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](image-url)
A.1.3 Design principle for NS deployment flavour

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

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.
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_3
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

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: b1bb8ce7-ebca-4fa7-95ed-4840d70a1177
        provider: MyCompany
The sunshineVNF.yaml file has the following content:

tosca_definitions_version: tosca_simple_yaml_1_3
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
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 0 ]
  - virtual_link_backend:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
  - virtual_link_service:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
interfaces:
  Vnflcm:
    type: tosca.interfaces.nfv.Vnflcm
    operations:
      instantiate:
        inputs:
          additional_parameters:
            type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters
            required: false

The vnf node template in the sunshine.vnfd.tosca.yaml file is abstract and is subject to substitution; the lower-level templates in the subsequent sections provide these substitutions. The actual lower-level template is selected based on the node type and a value constraint on the flavour_id property.

Each low level service template contains a node template of type MyCompany.SunshineDB.1_0.1_0 with implementation of the LCM interfaces.

SunshineDB (simple): Lower level
sunshinedbsimple.vnfd.tosca.yaml

This example illustrates one Vdu.Compute nodes (dbBackend) with two connection points and two virtual links (see figure A.2-2). The flavour_id is "Simple".

![Figure A.2-2: SunshineDB (simple): Lower level](image)
tosca_definitions_version: tosca_simple_yaml_1_3

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
  substitution_filter:
    properties:
      - flavour_id: { equal: 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:
        operations:
          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
  # ..
artifacts:
  sw_image:
    type: tosca.artifacts.nfv.SwImage
    file: maria.db.image.v1.0.qcow2
    properties:
      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

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 ]
SunshineDB (complex): Lower level

This example illustrates two Vdu.Compute nodes (dbBackend, and serviceNode) with two connection points and two virtual links (see figure A.2-3). The flavour_id is "complex".

NOTE: The single external VL above illustrates that both serviceNodeCp and dbBackendCp are connected to the same external VL. Alternatively, external connection points can be connected to separate external VLs.

Figure A.2-3: SunshineDB (complex): Lower level

sunshinedbcomplex.vnfd.tosca.yaml

tosca_definitions_version: tosca_simple_yaml_1_3

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
    substitution_filter:
        properties:
            - flavour_id: { equal: 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:
            operations:
                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
vdु_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
  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
      properties:
        name: Software of Maria Db
        version: '1.0'
        checksum:
          algorithm: sha-256
          hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664d3ed3cd74b4f9d
          container_format: bare
          disk_format: qcow2
          min_ram: 8192 MiB
          size: 2 GB
          operating_system: Linux
          supported_virtualisation_environments:
          - KVM

  mariaDbStorage:
    type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
    properties:
      virtual_block_storage_data:
        size_of_storage: 100 GB
        rdma_enabled: true
artifacts:
  sw_image:
    type: tosca.artifacts.nfv.SwImage
    file: maria.db.image.v1.0.qcow2
    properties:
      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

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

tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - exampleVNF.yaml # contains the VNF node type definition

#...
substitution_mappings:
  node_type: tosca.nodes.nfv.exampleVNF
  substitution_filter:
    properties:
      - flavour_id: { equal: simple }
    requirements:
      virtual_link: [vduCp_B2, virtual_link]

node_templates:
#..
  ExampleVNF:
    type: tosca.nodes.nfv.exampleVNF
    properties:
      flavour_description: A simple flavour
    interfaces:
      Vnflcm:
        operations:
          instantiate:
            implementation: instantiate.workbook.mistral.yaml
          terminate:
            implementation: terminate.workbook.mistral.yaml
  vduCp_B2:
    type: tosca.nodes.nfv.VduCp
    properties:
      layer_protocols: [ ipv4 ]
    # other 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.):
VDU-A:
  type: tosca.nodes.nfv.Vdu.Compute
  properties:
    name: VDU-A
    description: VDU-A description
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
    capabilities:
      virtual_compute:
        properties:
          virtual_memory:
            virtual_mem_size: 1 GB
          virtual_cpu:
            num_virtual_cpu: 1

vduCp_A:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: true
  requirements:
    - virtual_binding: VDU-A
    - virtual_link: intVL

intVL:
  type: tosca.nodes.nfv.VnfVirtualLink
  properties:
    connectivity_type:
      layer_protocols: [ ipv4 ]
    vl_profile:
      max_bitrate_requirements:
        root: 1000000
        leaf: 10000
      min_bitrate_requirements:
        root: 100000
        leaf: 10000

vduCp_B:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    protocol: [ associated_layer_protocol: ipv4 ]
    trunk_mode: true
  requirements:
    - virtual_binding: VDU-B
    - virtual_link: intVL

VDU-B:
  type: tosca.nodes.nfv.Vdu.Compute
  properties:
    name: VDU-B
    description: VDU-B description
    vdu_profile:
min_number_of_instances: 1
max_number_of_instances: 1
capabilities:
  virtual_compute:
    properties:
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1

The following snippet shows the relevant part of the file containing the imported node type definition:

tosca_definitions_version: tosca_simple_yaml_1_3
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:
      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: ExampleVNF ]
        default: ExampleVNF
      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' ]
# other properties omitted for brevity
requirements:
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](image)

The following snippet shows the corresponding node template definition.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - exampleVNF.yaml # contains the VNF node type definition

topology_template:
  #...

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    substitution_filter:
      properties:
        - flavour_id: { equal: simple }
      requirements:
        virtual_link: [myMRFExtCp, external_virtual_link]

  node_templates:
    ExampleVNF:
      type: tosca.nodes.nfv.exampleVNF
      properties:
```

- virtual_link:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo

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](image)

The following snippet shows the corresponding node template definition.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - exampleVNF.yaml # contains the VNF node type definition

topology_template:
  #...

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    substitution_filter:
      properties:
        - flavour_id: { equal: simple }
      requirements:
        virtual_link: [myMRFExtCp, external_virtual_link]

  node_templates:
    ExampleVNF:
      type: tosca.nodes.nfv.exampleVNF
      properties:
```

- virtual_link:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo

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](image)

The following snippet shows the corresponding node template definition.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - exampleVNF.yaml # contains the VNF node type definition

topology_template:
  #...

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    substitution_filter:
      properties:
        - flavour_id: { equal: simple }
      requirements:
        virtual_link: [myMRFExtCp, external_virtual_link]

  node_templates:
    ExampleVNF:
      type: tosca.nodes.nfv.exampleVNF
      properties:
```

- virtual_link:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo
flavour_description: A simple flavour
interfaces:
Vnflcm:
operations:
instantiate:
  implementation: instantiate.workbook.mistral.yaml
terminate:
  implementation: terminate.workbook.mistral.yaml

myMRFExtCp:
type: tosca.nodes.nfv.VnfExtCp
properties:
  layer_protocols: [ ipv4 ]
# other 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: 10000
    min_bitrate_requirements:
      root: 10000
      leaf: 1000
  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.
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_3
description: Template of a VNFD example
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - MyGroups.yaml #contains the node types definitions

topology_template:

  node_templates:
    Group_VDU_A:
      type: tosca.nodes.nfv.groupVDU_A
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 1 GB
            virtual_cpu:
              num_virtual_cpu: 1
          requirements:
            - virtual_link: VL_2

    Group_VDU_B:
      type: tosca.nodes.nfv.groupVDU_B
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 1 GB
            virtual_cpu:
```
num_virtual_cpu: 1
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.
capabilities:
  virtual_compute:
    properties:
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1
    requirements:
      - virtual_link: VL_1

VL_1:
type: tosca.nodes.nfv.VnfVirtualLink
properties:
  connectivity_type:
    layer_protocols: [ ipv4 ]
  vl_profile:
    max_bitrate_requirements:
      root: 1000000
      leaf: 100000
    min_bitrate_requirements:
      root: 100000
      leaf: 10000

# other properties omitted here for brevity

VL_2:
type: tosca.nodes.nfv.VnfVirtualLink
properties:
  connectivity_type:
    layer_protocols: [ ipv4 ]
  vl_profile:
    max_bitrate_requirements:
      root: 1000000
      leaf: 100000
    min_bitrate_requirements:
      root: 100000
      leaf: 10000

# other 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

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
  - MyGroups.yaml # contains the node types definitions

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:
      name: vduC_compute
      description: vduC_compute ..
      vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
      # other properties omitted here for brevity
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 1 GB
            virtual_cpu:
              num_virtual_cpu: 1
          #     artifacts:
          #       sw_image:             # omitted here for brevity
    requirements:
      - virtual_storage: vduC_storage
```

© ETSI 2014. All rights reserved
vduC_storage:
  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    virtual_block_storage_data:
      size_of_storage: 2 GB
      # other properties omitted here for brevity

internalCpd:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ipv4 ]
    # protocol: [ associated_layer_protocol: ipv4 ]
    # trunk_mode: false
    # properties omitted here for brevity
  requirements:
    - virtual_link:
      - virtual_binding: vduC_compute

The following template fragment provides the group definitions used in the above examples.

tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of VNFD types as defined in NFV SOL 001 GS
node_types:
  tosca.nodes.nfv.groupVDU:
    description: Abstract group VDU.
    derived_from: tosca.nodes.Root
    capabilities:
      virtual_compute:
        description: Describes virtual compute resources capabilities.
        type: tosca.capabilities.nfv.VirtualCompute
      virtual_storage:
        description: Defines the capabilities of virtual_storage.
        type: tosca.capabilities.nfv.VirtualStorage
    requirements:
      - virtual_link:
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        node: tosca.nodes.nfv.VnfVirtualLink
        occurrences: [ 1, 1 ]
  tosca.nodes.nfv.groupVDU_A:
    description: Abstract group VDU A.
    derived_from: tosca.nodes.nfv.groupVDU
  tosca.nodes.nfv.groupVDU_B:
    description: Abstract group VDU B.
    derived_from: tosca.nodes.nfv.groupVDU
    requirements:
      - virtual_link1:
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        node: tosca.nodes.nfv.VnfVirtualLink
        occurrences: [ 1, 1 ]
  tosca.nodes.nfv.groupVDU_C:
    description: Abstract group VDU C.
    derived_from: tosca.nodes.nfv.groupVDU
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-v1.3 [20] 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)](image-url)
additional_vnf_configurable_properties:
  type: MyCompany.datatypes.nfv.AuxVnfAdditionalConfigurableProperties
  required: true

MyCompany.datatypes.nfv.AuxVnfAdditionalConfigurableProperties:
  derived_from: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties
  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
      operations:
        instantiate:
          inputs:
            additional_parameters:
              type: MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters
required: false

#terminate:
MyCompany.nodes.nfv.Vdu.Aux:
  derived_from: tosca.nodes.nfv.Vdu.Compute
  properties:
    configurable_properties:
      type: MyCompany.datatypes.nfv.Aux.VnfcConfigurableProperties
      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:
          operations:
            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
artifacts:
  sw_image:
    type: tosca.artifacts.nfv.SwImage
    file: maria.db.image.v1.0.qcow2
    properties:
      name: Software of Maria Db
      version: '1.0'
      checksum:
        algorithm: sha-256
        hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4fd9
container_format: bare
disk_format: qcow2
min_disk: 2 GB
min_ram: 8192 MiB
size: 2 GB
operating_system: Linux
supported_virtualisation_environments:
  - KVM
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, ethernet ]
  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:
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, where it has only one entry. For a uniform aspect, the example 1 shows that step_deltas is omitted, whereas the example 2 shows step_deltas is included as a list of entries.

```yaml
node_templates:
  vdu_1:
    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
```

**Example 1**

tosca_definitions_version: tosca_simple_yaml_1_3
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:
```

---

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

Zero point (“smallest size”)

initial_delta

increment

Initial_delta (smallest size)
type: tosca.nodes.nfv.Vdu.Compute
properties:
    name: ..
    description: ..
    vdu_profile:
        min_number_of_instances: 1
        max_number_of_instances: 5
    nfvi_constraints:
        key_1: value_1
        key_2: value_2
    capabilities:
        virtual_compute:
            properties:
                virtual_memory:
                    virtual_mem_size: 1 GB
                virtual_cpu:
                    num_virtual_cpu: 1

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:
            key_1: value_1
            key_2: value_2
        capabilities:
            virtual_compute:
                properties:
                    virtual_memory:
                        virtual_mem_size: 1 GB
                    virtual_cpu:
                        num_virtual_cpu: 1

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:
            key_1: value_1
            key_2: value_2
        capabilities:
            virtual_compute:
                properties:
                    virtual_memory:
                        virtual_mem_size: 1 GB
                    virtual_cpu:
                        num_virtual_cpu: 1

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:
  key_1: value_1
  key_2: value_2
capabilities:
  virtual_compute:
    properties:
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1

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
      call_proc:
        name: ..
        description: ..
        max_scale_level: 4

- 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.VduScalingAspectDeltas
  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:
```yaml
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
```
EXAMPLE 2: This example is same as the example 1, except for ScalingAspects, which shows step_deltas list of entries.
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.

![Complex scaling example with non-uniform delta](image)

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_3
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:
        key_1: value_1
        key_2: value_2
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 1 GB
            virtual_cpu:
              num_virtual_cpu: 1

    oam:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: ..
        description: ..
      vdu_profile:
        min_number_of_instances: 1
        max_number_of_instances: 1
      nfvi_constraints:
        key_1: value_1
        key_2: value_2
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 1 GB
            virtual_cpu:
              num_virtual_cpu: 1

    processing:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: ..
        description:..
      vdu_profile:
        min_number_of_instances: 4
        max_number_of_instances: 12
      nfvi_constraints:
        key_1: value_1
        key_2: value_2
capabilities:
  virtual_compute:
    properties:
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1

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:
    key_1: value_1
    key_2: value_2

capabilities:
  virtual_compute:
    properties:
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1

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.
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.

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

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_nsd_types.yaml  # all of TOSCA NSD types as defined in ETSI GS NFV-SOL 001
  - example_VNF_2.yaml
```
node_types:
sasca.nodes.nfv.exampleNS:
  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
      relationship: tosca.relationships.nfv.VirtualLinksTo
      #...
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.example_VNF2
    # 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
          NsVirtualLink_1:
            type: tosca.nodes.nfv.NsVirtualLink
            properties:
              vl_profile:
                min_bitrate_requirements:
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](image)

The following snippet shows the corresponding node template definition:

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol0101_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:
      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
    relationship: tosca.relationships.nfv.VirtualLinksTo
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:
      protocol: [ associated_layer_protocol: ipv4 ]
      trunk_mode: false
      layer_protocols: [ ipv4 ]
    # other properties omitted for brevity
    requirements:
      - internal_virtual_link: NsVirtualLink_1
    # - external_virtual_link: # map to virtual_link requirement of the NS node

NsVirtualLink_1:
  type: tosca.nodes.nfv.NsVirtualLink
  properties:
    vl_profile:
      min_bitrate_requirements:
        root: 100000
      max_bitrate_requirements:
        root: 200000
      connectivity_type:
        layer_protocols: [ ipv4 ]

  # Other node templates
  #...

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-v1.3 [20] 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.

```
tosca_definitions_version: tosca_simple_yaml_1_3

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_VNF_1.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_VNF_2.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
```

---

Figure A.8-1: example_NS (simple)
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
operations:
instantiate:
  inputs:
    additional_parameters:
      type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters
      required: false
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:
        operations:
          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, ethernet]
    flow_pattern: mesh
    vl_profile:
      max_bitrate_requirements:
        root: 1000
      min_bitrate_requirements:
        root: 1000
    virtual_link_protocol_data:
      - associated_layer_protocol: ethernet
        l2_protocol_data:
          name: nsBaseNetwork
          network_type: vlan
          segmentation_id: VLAN100
      - associated_layer_protocol: ipv4
        l3_protocol_data:
          name: mybaseSubnet
          ip_version: ipv4
          cidr: 192.168.0.0/24
  policies:
    - my_service_NsMonitoring:
      type: tosca.policies.nfv.NsMonitoring
      properties:
        ns_monitoring_parameters:
          ns_monitoring_1111: #key map to the id of this monitoring parameter
          name: MyService_byte_incoming_sap
          performance_metric: byte_incoming_sap
          collection_period: 1 s
          ns_monitoring_2222: #key map to the id of this monitoring parameter
          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:
vnf_1_monitoring_1111: #key map to the id of this monitoring parameter
  name: VNF_1_v_cpu_usage_mean_vnf
  performance_metric: v_cpu_usage_mean_vnf
  collection_period: 1 s
vnf_1_monitoring_2222: #key map to the id of this monitoring parameter
  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:
      vnf_2_monitoring_1111: #key map to the id of this monitoring parameter
        name: VNF_2_v_cpu_usage_mean_vnf
        performance_metric: v_cpu_usage_mean_vnf
        collection_period: 1 s
      vnf_2_monitoring_2222: #key map to the id of this monitoring parameter
        name: VNF_2_v_disk_usage_mean_vnf
        performance_metric: v_disk_usage_mean_vnf
        collection_period: 1 s
      vnf_2_monitoring_3333: #key map to the id of this monitoring parameter
        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>
<thead>
<tr>
<th>ETSI GS NFV-IFA 011 [1] information element</th>
<th>TOSCA concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNFD (clause 7.1.2)</td>
<td>TOSCA service template(s) in the VNF package</td>
</tr>
<tr>
<td>VnfExtCpd (clause 7.1.4)</td>
<td>node template with type tosca.nodes.nfv.VnfExtCp or tosca.nodes.nfv.VduCp or tosca.nodes.nfv.VipCp</td>
</tr>
<tr>
<td>ETSI GS NFV-IFA 011 [1] information element</td>
<td>TOSCA concept</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VnfLcmOperationsConfiguration (clause 7.1.5)</td>
<td>property of VNF node type with data type tosca.datatypes.nfv.VnfLcmOperationsConfiguration and/or inputs additional parameters of the corresponding operation in the VnfLcm interface with type tosca.datatypes.nfv.VnfOperationAdditionalParameters</td>
</tr>
<tr>
<td>VLD (clause 7.1.7)</td>
<td>node template with type tosca.nodes.nfv.VnfVirtualLink</td>
</tr>
<tr>
<td>DeploymentFlavour (clause 7.1.8)</td>
<td>lower level service template(s) in the VNF package</td>
</tr>
<tr>
<td>VduProfile (clause 7.1.8.3)</td>
<td>property of Vdu.Compute node type with data type tosca.datatypes.nfv.VduProfile</td>
</tr>
<tr>
<td>VirtualLinkProfile (clause 7.1.8.4)</td>
<td>property of VnfVirtualLink node type with data type tosca.datatypes.nfv.VlProfile</td>
</tr>
<tr>
<td>VirtualLinkDescFlavour (clause 7.1.8.5)</td>
<td>Only qos attribute in VirtualLinkDescFlavour has been defined as property of the VlProfile data type</td>
</tr>
<tr>
<td>InstantiationLevel (clause 7.1.8.7)</td>
<td>policy with type tosca.policies.nfv.InstantiationLevels</td>
</tr>
<tr>
<td>VduLevel (clause 7.1.8.9)</td>
<td>policy with type tosca.policies.nfv.VduInstantiationLevels</td>
</tr>
<tr>
<td>LocalAffinityOrAntiAffinityRule (clause 7.1.8.11)</td>
<td>policy with type tosca.policies.nfv.AffinityRule or tosca.policies.nfv.AntiAffinityRule</td>
</tr>
<tr>
<td>AffinityOrAntiAffinityGroup (clause 7.1.8.12)</td>
<td>policy with type tosca.policies.nfv.AffinityRule or tosca.policies.nfv.AntiAffinityRule</td>
</tr>
<tr>
<td>Dependencies (clause 7.8.1.19)</td>
<td>dependency requirements between different node templates of type Vdu.Compute (or a derived node type)</td>
</tr>
<tr>
<td>VirtualComputeDesc (clause 7.1.9.2.2)</td>
<td>VirtualCompute capability of the Vdu.Compute node template</td>
</tr>
<tr>
<td>VirtualStorageDesc with BlockStorageData (clause 7.1.9.4.3)</td>
<td>node template with type tosca.nodes.nfv.Vdu.VirtualBlockStorage</td>
</tr>
<tr>
<td>VirtualStorageDesc with ObjectStorageData (clause 7.1.9.4.4)</td>
<td>node template with type tosca.nodes.nfv.Vdu.VirtualObjectStorage</td>
</tr>
<tr>
<td>VirtualStorageDesc with FileStorageData (clause 7.1.9.4.5)</td>
<td>node template with type tosca.nodes.nfv.Vdu.VirtualFileStorage</td>
</tr>
<tr>
<td>ScalingAspect (clause 7.1.10.2)</td>
<td>policy with type tosca.policies.nfv.ScalingAspects</td>
</tr>
<tr>
<td>ScalingDelta (clause 7.1.10.4)</td>
<td>policy with type tosca.policies.nfv.VduScalingAspectDeltas and/or tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas</td>
</tr>
<tr>
<td>VnfIndicator (clause 7.1.11.2)</td>
<td>interface with type tosca.interfaces.nfv.VnfIndicator and related attributes of the VNF node type.</td>
</tr>
<tr>
<td>MonitoringParameter (clause 7.1.11.3)</td>
<td>property of the VNF, Vdu.Compute and VnfVirtualLink node types</td>
</tr>
<tr>
<td>VnfConfigurableProperties (clause 7.1.12)</td>
<td>property of VNF node template with data type tosca.datatypes.nfv.VnfConfigurableProperties</td>
</tr>
<tr>
<td>LifeCycleManagementScript (clause 7.1.13)</td>
<td>interface with type tosca.interfaces.nfv.Vnflcm or tosca.interfaces.nfv.ChangeCurrentVnfPackage</td>
</tr>
</tbody>
</table>
### 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>
<thead>
<tr>
<th>ETSI GS NFV-IFA 014 [2] information element</th>
<th>TOSCA concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSD (clause 6.2.2)</td>
<td>TOSCA service template(s) in the NSD file structure</td>
</tr>
<tr>
<td>Sapd (clause 6.2.3)</td>
<td>node template with type tosca.nodes.nfv.Sap or the virtual_link requirement in the substitution_mapping</td>
</tr>
<tr>
<td>MonitoredData (clause 6.2.6)</td>
<td>policy with type tosca.policies.nfv.VnfMonitoring and tosca.policies.nfv.NsMonitoring</td>
</tr>
<tr>
<td>VnfIndicatorData (clause 6.2.7)</td>
<td>interface with type tosca.interfaces.nfv.NsVnfIndicator and related attributes of the NS node type</td>
</tr>
<tr>
<td>LifeCycleManagementScript (clause 6.2.9)</td>
<td>interface with type tosca.interfaces.nfv.Nslcm</td>
</tr>
<tr>
<td>VnfProfile (clause 6.3.3)</td>
<td>property of VNF node template with data type tosca.datatypes.nfv.VnfProfile</td>
</tr>
<tr>
<td>VirtualLinkProfile (clause 6.3.4)</td>
<td>property of NsVirtualLink node template with data type tosca.datatypes.nfv.VlProfile</td>
</tr>
<tr>
<td>AffinityOrAntiAffinityGroup (clause 6.3.5)</td>
<td>policy with type tosca.policies.nfv.NsAffinityRule or tosca.policies.nfv.NsAntiAffinityRule</td>
</tr>
<tr>
<td>NsVirtualLinkConnectivity (clause 6.3.7)</td>
<td>requirement and capability between PnfExtCp or VnfExtCp and NsVirtualLink node templates</td>
</tr>
<tr>
<td>NsLevel (clause 6.3.9)</td>
<td>policy with type tosca.policies.nfv.NsInstantiationLevels</td>
</tr>
<tr>
<td>NsProfile (clause 6.3.11)</td>
<td>property of NS node template with data type tosca.datatypes.nfv.NsProfile</td>
</tr>
<tr>
<td>Dependencies (clause 6.3.12)</td>
<td>dependency requirements</td>
</tr>
<tr>
<td>Vnffgd (clause 6.4.2)</td>
<td>group with type tosca.groups.nfv.VNFFG</td>
</tr>
<tr>
<td>Nfpd (clause 6.4.3)</td>
<td>node template with type tosca.nodes.nfv.NFP</td>
</tr>
<tr>
<td>NfpPositionElement (clause 6.4.6)</td>
<td>node template with type tosca.nodes.nfv.NfpPositionElement</td>
</tr>
<tr>
<td>NfpPositionDesc (clause 6.4.5)</td>
<td>node template with type tosca.nodes.nfv.NfpPosition</td>
</tr>
<tr>
<td>NsVirtualLinkDesc (clause 6.5.2)</td>
<td>node template with type tosca.nodes.nfv.NsVirtualLink</td>
</tr>
<tr>
<td>VirtualLinkDf (clause 6.5.4)</td>
<td>Only the serviceAvailabilityLevel 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.</td>
</tr>
<tr>
<td>Pnfd (clause 6.6.2)</td>
<td>A standalone TOSCA service template</td>
</tr>
<tr>
<td>PnfExtCpd (clause 6.6.4)</td>
<td>node template with type tosca.nodes.nfv.PnfExtCp</td>
</tr>
</tbody>
</table>
### A.10 PNFD modelling design example

In this example, TOSCA-Simple-Profile-YAML-v1.3 [20] 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.

```
tosca_definitions_version: tosca_simple_yaml_1_3

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
```

![Figure A.10-1: examplePnf](image)

<table>
<thead>
<tr>
<th>ETSI GS NFV-IFA 014 [2] information element</th>
<th>TOSCA concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>NsScalingAspect (clause 6.7.2)</td>
<td>policy with type tosca.policies.nfv.NsScalingAspects</td>
</tr>
<tr>
<td>VnfToLevelMapping (clause 6.7.4)</td>
<td>policy with type tosca.policies.nfv.VnfToLevelMapping and tosca.policies.nfv.VnfToInstantiationLevelMapping</td>
</tr>
<tr>
<td>VirtualLinkToLevelMapping (clause 6.7.5)</td>
<td>policy with type tosca.policies.nfv.VirtualLinkToLevelMapping and tosca.policies.nfv.VirtualLinkToInstantiationLevelMapping</td>
</tr>
<tr>
<td>NsToLevelMapping (clause 6.7.6)</td>
<td>policy with type tosca.policies.nfv.NsToLevelMapping and tosca.policies.nfv.NsToInstantiationLevelMapping</td>
</tr>
</tbody>
</table>
constraints: [ equal: an example PNF ]
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.

![NSD overview with multiple deployment flavour](image)

**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_3
description: my service
imports:
  - etsi_nfv_sol001_nsd_types.yaml
  - MyExampleNs_Type.yaml # contains the NS node type definition
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
directives:
  - substitute
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}
# requirements:
#  - virtual_link # mapped in lower-level templates

The MyExampleNs_Type.yaml file has the following content:

tosca_definitions_version: tosca_simple_yaml_1_3
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:
The NS node template in the myexample_NS.yaml file is abstract and is subject to substitution; the lower-level templates in the subsequent sections provide these substitutions. The actual lower-level template is selected based on the node type and a value constraint on the flavour_id property.

Each low level service template contains a node template of type 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_3
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
  substitution_filter:
    properties:
      - flavour_id: { equal: small }
    requirements:
      virtual_link: [ VNF_1, virtual_link_2 ]
node_templates:
  MyExampleNS:
    type: tosca.MyExampleNS
    # properties:
    #
    # interfaces:
    # Nslcm:
    #   operations:
    #     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.

MyExampleNs_big.yaml

tosca_definitions_version: tosca_simple_yaml_1_3

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

Figure A.11-3: MyExampleNs (big): Lower level
- 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
  substitution_filter:
    properties:
      - flavour_id: { equal: big }
    requirements:
      virtual_link: [ VNF_1, virtual_link_2 ]

node_templates:
  MyExampleNS:
    type: tosca.MyExampleNS
# # properties:
# #
  interfaces:
    NsLcm:
      operations:
        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_1: 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:
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
tosca_definitions_version: tosca_simple_yaml_1_3
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_TypeBis.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
    substitution_filter:
      properties:
        - flavour_id: { equal: small }
      requirements:
        virtual_link: [ VNF_1, virtual_link_2 ]

node_templates:
  NS_1:
    type: tosca.MyExampleNS
    interfaces:
      Nslcm:
        operations:
          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: c1bb0ab0-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
The contents of MyExampleNs_TypeBis.yaml file with the node type definition are as follows:

tosca_definitions_version: tosca_simple_yaml_1_3
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
        relationship: tosca.relationships.nfv.VirtualLinksTo
  interfaces:
    Nslcm:
      type: tosca.interfaces.nfv.Nslcm

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 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_3

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:
        operations:
          instantiate:
            implementation: instantiate.workflow.yaml
```
The following template fragment is part of the content in the etsi_nfv_example_vnf4.yaml. Template fragments for etsi_nfv_example_vnf1.yaml, etsi_nfv_example_vnf2.yaml and etsi_nfv_example_vnf3.yaml are available in clause A.14.

tosca_definitions_version: tosca_simple_yaml_1_3

description: Example VNF4 type

imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of VNFD types as defined in NFV SOL 001 GS

node_types:
tosca.nodes.nfv.example_VNF4:
derived_from: tosca.nodes.nfv.VNF
properties:
descriptor_id:
type: string
constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1184 ]
default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1184
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.
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.

```yaml
tosca_definitions_version: tosca_simple_yaml_1_3
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:
      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' ]
```
flavour_id:
  type: string
  constraints: [ valid_values: [ 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:
    operations:
      instantiate: {}  
      terminate: {} 

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:
    name: VDU-A
    description: VDU-A description
    vdu_profile:
      min_number_of_instances: 2
      max_number_of_instances: 5
    # other properties omitted for brevity
  capabilities:
    virtual_compute:
      properties: # following properties are required.
          virtual_memory:
            virtual_mem_size: 1 GB
          virtual_cpu:
            num_virtual_cpu: 1
    # requirements:
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.
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_3
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 VNF

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 ] ]
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:
# 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:
  name: VDU-A
description: VDU-A description
vdub_profile:
  
  min_number_of_instances: 1
  max_number_of_instances: 1
  # other properties omitted for brevity

capabilities:
  virtual_compute:
    properties: # following properties are required.
      virtual_memory:
        virtual_mem_size: 1 GB
      virtual_cpu:
        num_virtual_cpu: 1

  # requirements:
  # omitted for brevity

VDU-B:
  type: tosca.nodes.nfv.Vdu.Compute
  properties:
    name: VDU-B
    description: VDU-B description
  vdu_profile:
    min_number_of_instances: 1
    max_number_of_instances: 1
    # other properties omitted for brevity
  capabilities:
    virtual_compute:
      properties: # following properties are required.
        virtual_memory:
          virtual_mem_size: 1 GB
        virtual_cpu:
          num_virtual_cpu: 1

  # 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 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.
tosca_definitions_version: tosca_simple_yaml_1_3
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
  - 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

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
    operations:
      instantiate:
        inputs:
          additional_parameters:
            type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters
            required: false

requirements:
  - virtual_link:
      capability: tosca.capabilities.nfv.Virtuallinkable
      occurrences: [ 0, 0 ]
  - virtual_link_2:
      capability: tosca.capabilities.nfv.Virtuallinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
  - virtual_link_3:
      capability: tosca.capabilities.nfv.Virtuallinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
  - virtual_link_4:
      capability: tosca.capabilities.nfv.Virtuallinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]

topology_template:
  substitution_mappings:
    node_type: tosca.example_NS
  requirements:
    virtual_link 2: [ VNF_1_forward_1, 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_3, 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:
      operations:
        instantiate:
          implementation: instantiate.workflow.yaml
        terminate:
          implementation: terminateworkflow.yaml

NsVirtualLink_1:
  type: tosca.nodes.nfv.NsVirtualLink
  properties:
    vl_profile:
      min_bitrate_requirements:
        root: 100000
      max_bitrate_requirements:
        root: 200000
      connectivity_type:
        layer_protocols: [ ipv4 ]

VNF_1:
  type: tosca.nodes.nfv.example_VNF1
  properties:
    descriptor_id: b1bb0ce7-2222-4fa7-95ed-4840d70a1179
    descriptor_version: "1.0"
  requirements:
    - virtual_link: VNF_1_forward_1 # related to cp_01 in VNF_1
    - virtual_link_2: VNF_1_forward_2 # related to cp_02 in VNF_1

VNF_1_forward_1:
  type: tosca.nodes.nfv.Forwarding

VNF_1_forward_2:
  type: tosca.nodes.nfv.Forwarding
  requirements:
    - virtual_link: NsVirtualLink_1

VNF_2:
  type: tosca.nodes.nfv.example_VNF2
  properties:
    descriptor_id: b1bb0ce7-2222-4fa7-95ed-4840d70a1182
    descriptor_version: "1.0.0"
  requirements:
    - virtual_link: VNF_2_forward_3 # related to cp_03 in VNF_2
    - virtual_link_4: VNF_2_forward_4 # related to cp_04 in VNF_2

VNF_2_forward_3:
  type: tosca.nodes.nfv.Forwarding
VNF_2_forward_4:
  type: tosca.nodes.nfv.Forwarding
  requirements:
    - virtual_link: NsVirtualLink_1

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 # related to cp_05 in VNF_3
    - virtual_link_3: VNF_3_forward_6 # related to cp_06 in VNF_3

VNF_3_forward_5:
  type: tosca.nodes.nfv.Forwarding
  requirements:
    - virtual_link: NsVirtualLink_1

VNF_3_forward_6:
  type: tosca.nodes.nfv.Forwarding

# 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:
    forwardingBehaviour: lb
  requirements:
    - element: Element_1
    - element: Element_2

# NfpPosition_2 with Element_3 as constituents
NfpPosition_2:
  type: tosca.nodes.nfv.NfpPosition
  properties:
    forwardingBehaviour: 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:
        etherDestinationAddress: 00:0a:95:9d:68:16
        etherSourceAddress: 00:A0:C9:14:C8:29
        etherType: ipv4
        vlanTag:
          - "10"
          - "20"
          - "30"
        protocol: tcp
        dscp: "101111"
        sourcePortRange: [ 5000, 15000 ]
        destinationPortRange: [ 800, 8080 ]
        sourceIpAddressPrefix: 10.10.10.0
        destinationIpAddressPrefix: 125.1.12.111
        extendedCriteria:
          - startingPoint: 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 example_vnf1.yaml.

tosca_definitions_version: tosca_simple_yaml_1_3
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:
      descriptor_id:
        type: string
        constraints: [ equal: b1bb0ce7-2222-4fa7-95ed-4840d70a1179 ]
        default: b1bb0ce7-2222-4fa7-95ed-4840d70a1179
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: Example_VNF1 ]
        default: Example_VNF1
      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' ]
    requirements:
      - virtual_link_1:
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [ 0, 1 ]
      - virtual_link_2:
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [ 0, 1 ]
    topology_template:
      substitution_mappings:
        node_type: tosca.nodes.nfv.example_VNF1
      requirements:
        virtual_link: [ cp_01, external_virtual_link ]
        virtual_link_2: [ cp_02, external_virtual_link ]
    node_templates:
      cp_01:
        type: tosca.nodes.nfv.VnfExtCp
        properties:
          layer_protocols: [ ipv4 ]
requirements:
  - internal_virtual_link: intVL-A
cp_02:
  type: tosca.nodes.nfv.VnfExtCp
  properties:
    layer_protocols: [ ipv4 ]
requirements:
  - internal_virtual_link: intVL-A
intVL-A:
  type: tosca.nodes.nfv.VnfVirtualLink
  properties:
    connectivity_type:
      layer_protocols: [ ipv4 ]
    vl_profile:
      max_bitrate_requirements:
        root: 1000000
        leaf: 100000
      min_bitrate_requirements:
        root: 100000
        leaf: 10000

The following template fragment is part of the content in the example_vnf2.yaml.

tosca_definitions_version: tosca_simple_yaml_1_3
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:
    descriptor_id:
      type: string
      constraints: [ equal: b1bb0ce7-2222-4fa7-95ed-4840d70a1182 ]
      default: b1bb0ce7-2222-4fa7-95ed-4840d70a1182
    provider:
      type: string
      constraints: [ equal: MyCompany ]
      default: MyCompany
    product_name:
      type: string
      constraints: [ equal: Example_VNF2 ]
      default: Example_VNF2
    software_version:
      type: string
      constraints: [ equal: '1.0.0' ]
      default: '1.0.0'
    descriptor_version:
      type: string
      constraints: [ equal: '1.0.0' ]
      default: '1.0.0'
    flavour_id:
      type: string
      constraints: [ equal: simple ]
      default: simple
    flavour_description:
**vnfm_info:**
  - type: string
    default: ""
  - default: \[ '0:MyCompany-1.0.0' \]

**requirements:**
- virtual_link_1:
  - capability: tosca.capabilities.nfv.VirtualLinkable
  - relationship: tosca.relationships.nfv.VirtualLinksTo
  - occurrences: \[ 0, 1 \]
- virtual_link_2:
  - capability: tosca.capabilities.nfv.VirtualLinkable
  - relationship: tosca.relationships.nfv.VirtualLinksTo
  - occurrences: \[ 0, 1 \]
- virtual_link_4:
  - capability: tosca.capabilities.nfv.VirtualLinkable
  - relationship: tosca.relationships.nfv.VirtualLinksTo
  - occurrences: \[ 0, 1 \]

**topology_template:**

**node_type:**
  - tosca.nodes.nfv.example_VNF2

**node_templates:**
- cp_03:
  - type: tosca.nodes.nfv.VnfExtCp
  - properties:
    - layer_protocols: \[ ipv4 \]
  - requirements:
    - internal_virtual_link: intVL-A

- cp_04:
  - type: tosca.nodes.nfv.VnfExtCp
  - properties:
    - layer_protocols: \[ ipv4 \]
  - requirements:
    - internal_virtual_link: intVL-A

- intVL-A:
  - type: tosca.nodes.nfv.VnfVirtualLink
  - properties:
    - connectivity_type:
      - layer_protocols: \[ ipv4 \]
    - vl_profile:
      - max_bitrate_requirements:
        - root: 1000000
        - leaf: 10000
      - min_bitrate_requirements:
        - root: 100000
        - leaf: 10000
The following template fragment is part of the content in the example_vnf3.yaml.

tosca_definitions_version: tosca_simple_yaml_1_3
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:
descriptor_id:
type: string
  constraints: [ equal: b1bb0ce7-2222-4fa7-95ed-4840d70a1177 ]
  default: b1bb0ce7-2222-4fa7-95ed-4840d70a1177
provider:
type: string
  constraints: [ equal: MyCompany ]
  default: MyCompany
product_name:
type: string
  constraints: [ equal: Example_VNF3 ]
  default: Example_VNF3
software_version:
type: string
  constraints: [ equal: '1.0.0' ]
  default: '1.0.0'
descriptor_version:
type: string
  constraints: [ equal: '1.0.0' ]
  default: '1.0.0'
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']
requirements:
  - virtual_link_1:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
  - virtual_link_2:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
  - virtual_link_3:
      capability: tosca.capabilities.nfv.VirtualLinkable
      relationship: tosca.relationships.nfv.VirtualLinksTo
      occurrences: [ 0, 1 ]
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:
      layer_protocols: [ ipv4 ]
    # . . .
    requirements:
      - internal_virtual_link: intVL-A
  cp_06:
    type: tosca.nodes.nfv.VnfExtCp
    properties:
      layer_protocols: [ ipv4 ]
    # . . .
    requirements:
      - internal_virtual_link: intVL-A
intVL-A:
    type: tosca.nodes.nfv.VnfVirtualLink
    properties:
      connectivity_type:
        layer_protocols: [ ipv4 ]
      vl_profile:
        max_bitrate_requirements:
          root: 1000000
          leaf: 10000
        min_bitrate_requirements:
          root: 10000
          leaf: 1000

A.15 Auto-scale and auto-heal design

A.15.1 Introduction

This clause describes the Auto-scale and auto-heal policy design in a VNFD or NSD.

A.15.2 Auto-scale and auto-heal design with use of VNF indicator in VNFD

This clause illustrates an example of a VNF with two VNF indicators, one VNF indicator related to the utilization of the VNF and one related to the VNF health.

The modelling and handling of the indicator in the VNFD service template involves the following definitions:

- A VNF specific VNF indicator interface type definition that includes one notification for each VNF indicator.
- New attribute definition included in the VNF specific node type definition for each of the indicators.
- Two attributes to hold the values of the current scale level of the 'call_proc' and 'database' scaling aspects.
NOTE: It is assumed that this VNF has the same scaling aspects as defined in the example in clause A.6 ('call_proc' and 'database').

- An interface definition of the VNF specific Vnflcm type with input definitions of additional_parameters for the heal operation.
- An interface definition of the VNF specific VNF indicator type with an output definition for each notification indicating the attribute modified by the output of the notification.
- A VNF specific auto_scale and auto_heal policy definitions of tosca.nfv.policies.VnfIndicator type. Each policy includes a trigger with an event associated to the notification, as well as condition and action.

tosca_definitions_version: tosca_simple_yaml_1_3
imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

interface_types:
  MyCompany.nfv.interfaces.VnfIndicator:
    derived_from: tosca.interfaces.nfv.VnfIndicator
    notifications:
      utilization:
        description: this notification is used to received asynchronous information of value change of the utilization_vnf_indicator
      health:
        description: this notification is used to received asynchronous information of value change of the health_vnf_indicator

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 ] ]
        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' ]
attributes:
  utilization_vnf_indicator:
    type: float
    description: holds the value of the utilization VNF indicator. It is assigned the output value of the utilization notification.
  health_vnf_indicator:
    type: string
    description: holds the value of the health VNF indicator. It is assigned the output value of the health notification.
  call_proc_scale_level:
    type: integer
  database_scale_level:
    type: integer
interfaces:
  Vnflcm:
    operations:
      # scale: {}
    heal:
      inputs:
        additional_parameters:
          type: MyCompany.datatypes.nfv.HealAdditionalParameters
          required: false
    VnfIndicator:
      type: MyCompany.nfv.interfaces.VnfIndicator
      notifications:
        utilization:
          outputs:
            utilization_vnf_indicator: [ SELF, utilization_vnf_indicator ]
        health:
          outputs:
            health_vnf_indicator: [ SELF, health_vnf_indicator ]
data_types:
  MyCompany.datatypes.nfv.HealAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      parameter_1:
        type: string
        required: false
policy_types:
  MyCompany.policies.nfv.VnfIndicator:
    derived_from: tosca.policies.nfv.VnfIndicator
topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    # ...
  node_templates:
    SunshineDB:
      type: MyCompany.SunshineDB.1_0.1_0
      attributes:
        call_proc_scale_level: { get_attribute: [ SELF, scale_status, call_proc, scale_level ] }
database_scale_level: { get_attribute: [ SELF, scale_status, database, scale_level ] }

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
- auto_scale:
  type: tosca.policies.nfv.VnfIndicator
  properties:
    source: vnf
  triggers:
    scale_out:
      event: tosca.interfaces.nfv.VnfIndicator.utilization # full name of a notification in the VnfIndicator interface
    condition:
      - utilization_vnf_indicator: [ { greater_or_equal: 60.0 } ]
      - call_proc_scale_level: [ { less_than: 3 } ]
    action:
      - call_operation:
          operation: Vnflcm.scale
          inputs:
            type:
              value: scale_out
            aspect:
              value: call_proc
            number_of_steps:
              value: 1 # optional
    scale_in:
      event: tosca.interfaces.nfv.VnfIndicator.utilization # full name of a notification in the VnfIndicator interface
      condition:
        - utilization_vnf_indicator: [ { less_or_equal: 20.0 } ]
        - call_proc_scale_level: [ { greater_than: 0 } ]
      action:
        - call_operation:
            operation: Vnflcm.scale
            inputs:
              type:
                value: scale_in
              aspect:
                value: call_proc
      targets: [SunshineDB ]
- auto_heal:
  type: tosca.policies.nfv.VnfIndicator
  properties:
A.15.3 Auto-scale design with use of VNF indicator in NSD

This clause illustrates an example of a NS with two VNF indicators, one VNF indicator related to the utilization of the VNF_1 and one related to the utilization of the VNF_2.

The modelling and handling of the VNF indicator in the NSD service template involves the following definitions:

- A NS specific VNF indicator interface type definition that includes one notification for VNF indicators.
- New attribute definition included in the NS specific node type definition for each of the VNF indicators.
- Two attributes to hold the values of the current scale level of the 'call_proc' and 'database' scaling aspects.
- A NS specific auto_scale policy definitions of tosca.policies.nfv.NsAutoScale type, which includes a trigger with an event associated to the notification, as well as condition and action.

Example_NS.yaml:

tosca_definitions_version: tosca_simple_yaml_1_3

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_VNF_1.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_VNF_2.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

interface_types:
  MyCompany.nfv.interfaces.NsVnfIndicator:
    derived_from: tosca.interfaces.nfv.NsVnfIndicator
    notifications:
      utilization:
        description: this notification is used to received asynchronous information of value change of the utilization_vnf_indicator

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
attributes:
vnf_1_utilization_vnf_indicator:
type: float
description: holds the value of the vnf_1_utilization VNF indicator. It is assigned the output value of the utilization notification.
vnf_2_utilization_vnf_indicator:
type: float
description: holds the value of the vnf_1_utilization VNF indicator. It is assigned the output value of the utilization notification.
call_proc_scale_level:
type: integer
database_scale_level:
type: integer
interfaces:
Nslcm:
type: tosca.interfaces.nfv.Nslcm
operations:
  instantiate:
    inputs:
      additional_parameters:
        type: MyCompany.datatypes.nfv.NsInstantiateAdditionalParameters
        required: false
NsVnfIndicator:
type: MyCompany.nfv.interfaces.NsVnfIndicator
notifications:
  utilization: {}

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
attributes:
call_proc_scale_level: { get_attribute: [ SELF, scale_status, call_proc ] }
vnf_1_utilization_vnf_indicator: { get_attribute: [ VNF_1, utilization_vnf_indicator ] } # required Vnf indicator for VNF_1 of this NS
vnf_2_utilization_vnf_indicator: { get_attribute: [ VNF_2, utilization_vnf_indicator ] } # required Vnf indicator for VNF_1 of this NS
interfaces:
  Nslcm:
    operations:
      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:
    - scaling_aspects:
        type: tosca.policies.nfv.NsScalingAspects
        properties:
          aspects:
            database:
              name: DatabaseScalingAspect
              description: This scaling aspect is for database
              ns_scale_levels:
                0:
                  description: first NS level for the database scaling aspect
1:
   description: second NS level for the database scaling aspect
call_proc:
   name: ProcessorScalingAspect
   description: This scaling aspect is for Processor
ns_scale_levels:
  0:
   description: first NS level for the processor scaling aspect
  1:
   description: second NS level for the processor scaling aspect
  2:
   description: third NS level for the processor scaling aspect
- auto_scale:
  type: tosca.policies.nfv.NsAutoScale
  # properties:
  # none
  triggers:
  scale_out:
   event: tosca.interfaces.nfv.NsVnfIndicator.utilization # full name of a notification in the VnfIndicator interface
   condition:
     - vnf_1_utilization_vnf_indicator: [ { greater_or_equal: 80.0 } ]
     - vnf_2_utilization_vnf_indicator: [ { greater_or_equal: 80.0 } ]
     - call_proc_scale_level: [ { less_than: 2 } ]
   action:
     - call_operation:
       operation: Nslcm.scale
       inputs:
         scale_ns_by_steps_data:
           scaling_direction: scale_out
           aspect: call_proc
           number_of_steps: 1 # optional
  scale_in:
   event: tosca.interfaces.nfv.NsVnfIndicator.utilization # full name of a notification in the VnfIndicator interface
   condition:
     - vnf_1_utilization_vnf_indicator: [ { less_or_equal: 10.0 } ]
     - vnf_2_utilization_vnf_indicator: [ { less_or_equal: 10.0 } ]
     - call_proc_scale_level: [ { greater_than: 2 } ]
   action:
     - call_operation:
       operation: Nslcm.scale
       inputs:
         scale_ns_by_steps_data:
           scaling_direction: scale_out
           aspect: call_proc
           number_of_steps: 1 # optional

   targets: [ my_service ]

The contents of example_VNF_1.yaml file with the VNF_1 node type definition are as follows:

tosca_definitions_version: tosca_simple_yaml_1_3
description: VNFD for VNF_1

imports:
  - etsi_nfv_sol001_vnfd_types.yaml # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001
interface_types:
    MyCompany.nfv.interfaces.VnfIndicator:
        derived_from: tosca.interfaces.nfv.VnfIndicator
    notifications:
        utilization:
            description: this notification is used to received asynchronous information of value change of the utilization_vnf_indicator
        health:
            description: this notification is used to received asynchronous information of value change of the health_vnf_indicator

node_types:
    tosca.nodes.nfv.example_VNF1:
        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 ] ]
                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' ]
        attributes:
            utilization_vnf_indicator:
                type: float
                description: holds the value of the utilization VNF indicator. It is assigned the output value of the utilization notification.
            health_vnf_indicator:
                type: string
                description: holds the value of the health VNF indicator. It is assigned the output value of the health notification.
            call_proc_scale_level:
                type: integer
            database_scale_level:
The contents of example_VNF_2.yaml file with the VNF_2 node type definition are as follows:

tosca_definitions_version: tosca_simple_yaml_1_3
description: VNFD for VNF_2

imports:
  - etsi_nfv_sol001_vnfd_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

interface_types:
  MyOtherCompany.nfv.interfaces.VnfIndicator:
    derived_from: tosca.interfaces.nfv.VnfIndicator
    notifications:
      utilization:
        description: this notification is used to received asynchronous information of value change of the utilization_vnf_indicator
      health:
        description: this notification is used to received asynchronous information of value change of the health_vnf_indicator

node_types:
  tosca.nodes.nfv.example_VNF2:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
        type: string
        constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233 ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a2233
      provider:
        type: string
        constraints: [ equal: MyOtherCompany ]
        default: MyOtherCompany
      product_name:
        type: string
        constraints: [ equal: MyProduct ]
        default: MyProduct
      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 ] ]
  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' ]
attributes:
  utilization_vnf_indicator:
    type: float
    description: holds the value of the utilization VNF indicator. It is assigned the output value of the utilization notification.
  health_vnf_indicator:
    type: string
    description: holds the value of the health VNF indicator. It is assigned the output value of the health notification.
  call_proc_scale_level:
    type: integer
  database_scale_level:
    type: integer
interfaces:
  Vnflcm:
    operations:
      scale: {}
  VnfIndicator:
    type: MyOtherCompany.nfv.interfaces.VnfIndicator
    notations:
      utilization:
        outputs:
          utilization_vnf_indicator: [ SELF, utilization_vnf_indicator ]
      health:
        outputs:
          health_vnf_indicator: [ SELF, health_vnf_indicator ]
requirements:
- virtual_link:
  capability: tosca.capabilities.nfv.VirtualLinkable
  occurrences: [ 0, 0 ]
- virtual_link_1:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo
  occurrences: [ 1, 1 ]
- virtual_link_2:
  capability: tosca.capabilities.nfv.VirtualLinkable
  relationship: tosca.relationships.nfv.VirtualLinksTo
  occurrences: [ 0, 1 ]

A.16 VDU connection point in trunk mode

With supporting trunk mode, a VM can differentiate between traffic of many networks by different encapsulation types and IDs, instead of using many vNICs. Connection points of a VDU are combined into one VIF by turning one connection point into a trunk parent port and the other connection points into trunk subports of the same trunk. A trunk logical model is a collection of connection points. Different infrastructure technologies may have different way to implement trunk logical model.
The main concept of difference between subport and parent port:

- The parent port can associate to the VM and a network.
- The parent port can be associated by subports.
- The subport can not associate directly to the VM but through the parent port and a network.
- The traffic of sub-port can be encapsulated as required.

In the following example, there are three connection points are needed in initial deployment stage, one VdpCp node taken as trunk port and two VduSubCp nodes taken as trunk subports. With respect to VduSubCp node, requirement virtual\_binding, if existing, is useless because the trunk\_binding have priority over it.

Figure A.16-1: VNFD with a VDU has three connection points group as trunk

The following service template shows the relevant parts of the TOSCA VNFD corresponding to figure A.16-1:

```
tosca_definitions_version: tosca_simple_yaml_1_3
# ...
imports:
  - etsi_nfv_sol001_vnf_types.yaml  # all of TOSCA VNFD types as defined in ETSI GS NFV-SOL 001

topology_template:
  node_templates:
    #...
    VduwithTrunk:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: VduwithTrunk
        description: compute node with trunk port
        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:
```

num_virtual_cpu: 2
# omitted for brevity

TrunkParentPort:
  type: tosca.nodes.nfv.VduCp
  properties:
  # omitted for brevity
  layer_protocols: [ ipv4 ]
  trunk_mode: true
requirements:
  - virtual_binding: VduwithTrunk
  - virtual_link: intVL1

TrunkSubPort1:
  type: tosca.nodes.nfv.VduSubCp
  properties:
  # omitted for brevity
  layer_protocols: [ ipv4 ]
  segmentation_type: vlan
  segmentation_id: 10
requirements:
  - trunk_binding: TrunkParentPort
  - virtual_link: intVL2

TrunkSubPort2:
  type: tosca.nodes.nfv.VduSubCp
  properties:
  # omitted for brevity
  layer_protocols: [ ipv4 ]
  segmentation_type: vlan
  segmentation_id: 100
requirements:
  - trunk_binding: TrunkParentPort
  - virtual_link: intVL3

intVL1:
  type: tosca.nodes.nfv.VnfVirtualLink
  properties:
  connectivity_type:
  # omitted for brevity
  layer_protocols: [ ipv4 ]
  vl_profile:
    max_bitrate_requirements:
      root: 100000
      leaf: 20000
    min_bitrate_requirements:
      root: 10000
      leaf: 10000 # omitted for brevity

intVL2:
  type: tosca.nodes.nfv.VnfVirtualLink
  properties:
  connectivity_type:
  # omitted for brevity
  layer_protocols: [ ipv4 ]
  vl_profile:
    max_bitrate_requirements:
      root: 100000
      leaf: 20000
    min_bitrate_requirements:
      root: 10000
      leaf: 10000

# omitted for brevity

intVL3:
A.17 NS scaling

Clause A.17 shows the use of the NS policies to define the scaling aspects and NS scale levels.

The NSD NS_1 consists of the following components:

- Three VNFDs specified with the VNF node templates: FrontEnd, CentralProcess and Database
- One nested NSD specified with the NS node template: SupportFrontEnd
- One NsVirtualLinkDesc specified with the NS VL node template: NS_VL_1.

Figure A.17-1 shows the two scaling aspects and the two instantiation levels defined in this NS.

![NS scaling aspects and instantiation levels](image)

tosca_definitions_version: tosca_simple_yaml_1_3
description: myExampleNs with scaling aspects
imports:
  - etsi_nfv_sol001_nsd_types.yaml  # all of TOSCA NSD types as defined in ETSI GS NFV-SOL 001
  - MyScalableNs_Type.yaml # contains the NS node type definition.
- SupportFrontEndNs.yaml # uri of the yaml file which contains the
  MyProviderCompany.NS.SupportFrontEndNs node type definition, this file might be
  included in the NSD file structure of NS_1.
- FrontEnd_1_0.yaml # uri of the yaml file which contains the
  MyCompany.VNF.FrontEnd_1_0 node type definition, this file might be included in the NSD
  file structure of NS_1.
- Database_1_0.yaml # uri of the yaml file which contains the
  MyCompany.VNF.Database_1_0 node type definition, this file might be included in the NSD
  file structure of NS_1.
- CentralProcess_1_0.yaml # uri of the yaml file which contains the
  MyCompany.VNF.CentralProcess_1_0 node type definition, this file might be included in
  the NSD file structure of NS_1.

topology_template:
  substitution_mappings:
    node_type: MyProviderCompany.NS.MyScalableNS
  substitution_filter:
    properties:
      - flavour_id: { equal: small }
  requirements:
    virtual_link: [ FrontEnd, virtual_link_2 ]

node_templates:
  NS_1:
    type: MyProviderCompany.NS.MyScalableNS
    interfaces:
      Nslcm:
        operations:
          instantiate:
            implementation: instantiate.workflow.yaml
          terminate:
            implementation: terminate.workflow.yaml
  FrontEnd:
    type: MyCompany.VNF.FrontEnd_1_0
    properties:
      # no property assignments needed for required properties that have a default
      flavour_id: simple
      vnf_profile:
        instantiation_level: level_1
        min_number_of_instances: 1
        max_number_of_instances: 4
      requirements:
        - virtual_link_1: NS_VL_1
        # - virtual_link_2: # map to virtual_link requirement of the NS node
  Database:
    type: MyCompany.VNF.Database_1_0
    properties:
      # no property assignments needed for required properties that have a default
      flavour_id: simple
      vnf_profile:
        instantiation_level: level_1
        min_number_of_instances: 1
        max_number_of_instances: 2
      requirements:
        - virtual_link_1: NS_VL_1
  CentralProcess:
type: MyCompany.VNF.CentralProcess_1_0
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: 1
    max_number_of_instances: 2
  requirements:
    - virtual_link_1: NS_VL_1

SupportFrontEnd:
  type: MyProviderCompany.NS.SupportFrontEndNs
  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: 0
    max_number_of_instances: 1
    flavour_id: simple
    flavour_id: simple
  requirements:
    - virtual_link_1: NS_VL_1

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
  policies:
    - scaling_aspects:
        type: tosca.policies.nfv.NsScalingAspects
        properties:
          aspects:
            database:
              name: DatabaseScalingAspect
              description: This scaling aspect is used to scale the Database VNF
              ns_scale_levels:
                0:
                  description: first NS level for the database scaling aspect
                1:
                  description: second NS level for the database scaling aspect
            processor:
              name: ProcessorScalingAspect
              description: This scaling aspect is used to scale the FrontEnd and
              CentralProcess VNFs and the SupportFrontEnd nested NS
              ns_scale_levels:
                0:
                  description: first NS level for the processor scaling aspect
1:  
   description: second NS level for the processor scaling aspect
2:  
   description: second NS level for the processor scaling aspect

- FrontEndVnfToLevelMapping:
  type: tosca.policies.nfv.VnfToLevelMapping
  properties:
    aspect: processor
    number_of_instances:
      0: 1
      1: 3
      2: 4
  targets: [ FrontEnd ]

- CentralProcessVnfToLevelMapping:
  type: tosca.policies.nfv.VnfToLevelMapping
  properties:
    aspect: processor
    number_of_instances:
      0: 1
      1: 1
      2: 2
  targets: [ CentralProcess ]

- DatabaseVnfToLevelMapping:
  type: tosca.policies.nfv.VnfToLevelMapping
  properties:
    aspect: database
    number_of_instances:
      0: 1
      1: 2
  targets: [ Database ]

- SupportFrontEndNsToLevelMapping:
  type: tosca.policies.nfv.NsToLevelMapping
  properties:
    aspect: processor
    number_of_instances:
      0: 0
      1: 0
      2: 1
  targets: [ SupportFrontEnd ]

- ns_instantiation_levels:
  type: tosca.policies.nfv.NsInstantiationLevels
  properties:
    ns_levels:
      instantiation_level_1:
        description: low capacity instantiation level
      instantiation_level_2:
        description: high capacity instantiation level
    default_level: instantiation_level_1

- FrontEnd_instantiation_levels:
  type: tosca.policies.nfv.VnfToInstantiationLevelMapping
  properties:
    number_of_instances:
      instantiation_level_1: 1
      instantiation_level_2: 3
  targets: [ FrontEnd ]
- CentralProcess_instantiation_levels:
  type: tosca.policies.nfv.VnfToInstantiationLevelMapping
  properties:
    number_of_instances:
      instantiation_level_1: 1
      instantiation_level_2: 1
    targets: [ CentralProcess ]

- Database_instantiation_levels:
  type: tosca.policies.nfv.VnfToInstantiationLevelMapping
  properties:
    number_of_instances:
      instantiation_level_1: 1
      instantiation_level_2: 1
    targets: [ Database ]

- SupportFrontEnd_instantiation_levels:
  type: tosca.policies.nfv.NsToInstantiationLevelMapping
  properties:
    number_of_instances:
      instantiation_level_1: 0
      instantiation_level_2: 0
    targets: [ SupportFrontEnd ]

The contents of MyScalableNs_Type.yaml file with the node type definition are as follows:

tosca_definitions_version: tosca_simple_yaml_1_3
description: type definition of tosca.MyScalableNS

imports:
  - etsi_nfv_sol001_nsd_types.yaml  # all of TOSCA types as defined in ETSI GS NFV-SOL 001

node_types:
  MyProviderCompany.NS.MyScalableNS:
    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

A.18 VNFD illustrating OsContainer modeling example

tosca_definitions_version: tosca_simple_yaml_1_3
description: A sample VNF descriptor with containerized VDUs
imports:
  - etsi_nfv_sol001_vnfd_types.yaml # VNFD definitions based on SOL001 v4.2.1 draft
data_types:
  Example.datatypes.nfv.VnfAdditionalConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties
    properties:
      name_prefix_in_cism:
        type: string
        required: false
      dns_server:
        type: list
        entry_schema:
          type: string
          required: false
  Example.datatypes.nfv.VnfConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfConfigurableProperties
    properties:
      additional_configurable_properties:
        type: Example.datatypes.nfv.VnfAdditionalConfigurableProperties
node_types:
  ExampleCorp.vDB.0_1.0_2:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
        default: 5af09567-fc5f-4be9-b372-0cc431ad5c03
        constraints: [ equal: 5af09567-fc5f-4be9-b372-0cc431ad5c03 ]
      descriptor_version:
        default: "0.2"
        constraints: [ equal: "0.2" ]
      provider:
        default: "ExampleCorp"
        constraints: [ equal: "ExampleCorp" ]
      product_name:
        default: "vDB"
        constraints: [ equal: "vDB" ]
      software_version:
        default: "0.1"
        constraints: [ equal: "0.1" ]
      vnfm_info:
        type: list
required: true
description: Identifies VNFM(s) compatible with the VNF
type: string
entry_schema:
  - equal: 'etsivnfm:v0.1.0'
default: ['etsivnfm:v0.1.0']
flavour_id:
default: simple
constraints:
  - valid_values: [ simple, complex ]
flavour_description:
default: ""
configurable_properties:
type: Example.datatypes.nfv.VnfConfigurableProperties
description: Describes the configurable properties of the VNF
required: false
requirements:
  - virtual_link:
    capability: tosca.capabilities.nfv.VirtualLinkable
    occurrences: [0, 0]
  - virtual_link_1:
    capability: tosca.capabilities.nfv.VirtualLinkable
    relationship: tosca.relationships.nfv.VirtualLinksTo
    occurrences: [0, 1]
  - virtual_link_2:
    capability: tosca.capabilities.nfv.VirtualLinkable
    relationship: tosca.relationships.nfv.VirtualLinksTo
    occurrences: [0, 1]
  - virtual_link_virt:
    capability: tosca.capabilities.nfv.VirtualLinkable
    relationship: tosca.relationships.nfv.VirtualLinksTo
    occurrences: [0, 1]
  - virtual_link_intra_1:
    capability: tosca.capabilities.nfv.VirtualLinkable
    relationship: tosca.relationships.nfv.VirtualLinksTo
    occurrences: [0, 1]
  - virtual_link_intra_2:
    capability: tosca.capabilities.nfv.VirtualLinkable
    relationship: tosca.relationships.nfv.VirtualLinksTo
    occurrences: [0, 1]

topology_template:
substitution_mappings:
  node_type: ExampleCorp.vDB.0_1.0_2
requirements:
  virtual_link_1: [ vdu1Cp1, virtual_link ]
  virtual_link_2: [ vdu2Cp1, virtual_link ]
  virtual_link_virt: [ VirtCp1, virtual_link ]
  virtual_link_intra_1: [ vdu1Cp2, virtual_link ]
  virtual_link_intra_2: [ vdu2Cp2, virtual_link ]

node_templates:
VNF:
type: ExampleCorp.vDB.0_1.0_2
properties:
  product_info_name: "Firewall"
  product_info_description: "Firewall with new functionality"
configurable_properties: # All of the VNF configurable properties should
go here
  is_autoscale_enabled: true
  is_autoheal_enabled: true
additional_configurable_properties:
  name_prefix_in_cism: "ExampleContainerizedVNF" # Default
dns_server: [ "8.8.8.8", "8.8.4.4" ] # Default

# modifiable_attributes: {} # All of the VNF modifiable properties should go here
flavour_id: simple
flavour_description: "A specimen copy of the VNF Descriptor for containers"

interfaces:
  Vnflcm:
    operations:
      # instantiate_start:
      instantiate: # A sample illustration to run a instantiation script without any inputs.
      instantiate_end: helm_test

    scale: # Invoked upon receipt of a Scale VNF request
    inputs:
      type: "scale_out"
      aspect: "default"

    artifacts:
      helm_test:
        description: Post instantiation test script
type: tosca.artifacts.Implementation.Bash
file: Artifacts/Tests/helm_test.sh

opendb_mciop:
  type: tosca.nodes.nfv.Mciop
requirements:
  - associatedVdu: Vdu_2
artifacts:
  opendb_helm:
    description: Helm Chart for opendb Pod
type: tosca.artifacts.nfv.HelmChart
file: Artifacts/Scripts/opendb.tgz

lb_mciop:
  type: tosca.nodes.nfv.Mciop
requirements:
  - associatedVdu: Vdu_1
  - dependency: opendb_mciop
artifacts:
  lb_helm:
    description: Helm Chart for lb Pod
type: tosca.artifacts.nfv.HelmChart
file: Artifacts/Scripts/lb.tgz

vdu1Cp1:
  type: tosca.nodes.nfv.VduCp
description: "Cluster vNIC"
properties:
  layer_protocols: [ ethernet ]
  vnic_type: normal
  order: 1
requirements:
  - virtual_binding: Vdu_1

vdu1Cp2:
  type: tosca.nodes.nfv.VduCp
description: "IntraVNF External vNIC"
properties:
layer_protocols: [ ethernet ]
vnic_type: macvlan
bitrate_requirement: 1000000000 #10 GiB
order: 2
trunk_mode: true
requirements:
- virtual_binding: Vdu_1

VirtCp1:
type: tosca.nodes.nfv.VirtualCp
properties:
layer_protocols: [ ipv6 ]
protocol:
- address_data:
  13_address_data:
    floating_ip_activated: true
    ip_address_assignment: false
    ip_address_type: ipv6
    number_of_ip_address: 1
    associated_layer_protocol: ipv6
additionalServiceData:
- portData:
  - name: DBAccessService
    protocol: tcp
    port: 8001
    portConfigurable: false
  - name: DBConfigService
    protocol: tcp
    port: 443
    portConfigurable: false
requirements:
- target: Vdu_1

Vdu_1:
type: tosca.nodes.nfv.Vdu.OsContainerDeployableUnit
properties:
name: "lb"
description: "Advanced Software Load Balancer"
vdu_profile:
  min_number_of_instances: 1
  max_number_of_instances: 4
requirements:
- container: Vdu_1_Container_1

Vdu_1_Container_1:
type: tosca.nodes.nfv.Vdu.OsContainer
properties:
name: "lb Container"
description: "Advanced Software Load Balancer"
requested_cpu_resources: 100 # In Milli-Cpus, ie 0.1 CPU
cpu_resource_limit: 1000 # In Milli-Cpus, ie 1 CPU, single threaded
requested_memory_resources: 10 MiB
memory_resource_limit: 100 MiB
artifacts:
sw_image:
type: tosca.artifacts.nfv.SwImage
file: Artifacts/Images/lbimage
properties:
name: lb
version: "7.3"
checksum:
  algorithm: sha-256
  hash: a411cafee2f0f702572369da0b765e2
container_format: docker
size: "1024MB"

vdu2Cp1:
  type: tosca.nodes.nfv.VduCp
  properties:
    layer_protocols: [ ethernet ]
    vnic_type: normal
  requirements:
    - virtual_binding: Vdu_2

vdu2Cp2:
  type: tosca.nodes.nfv.VduCp
  description: "IntraVNF External vNIC"
  properties:
    layer_protocols: [ ethernet ]
    vnic_type: macvlan
    bitrate_requirement: 10000000000 #10 GiB
    order: 2
    trunk_mode: true
  requirements:
    - virtual_binding: Vdu_2

opendbDbStorage:
  type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
  properties:
    virtual_block_storage_data:
      size_of_storage: 100 GB
      rdma_enabled: false

Vdu_2:
  type: tosca.nodes.nfv.Vdu.OsContainerDeployableUnit
  properties:
    name: "opendb"
    description: "DB Server"
    vdu_profile:
      min_number_of_instances: 1
      max_number_of_instances: 1
  requirements:
    - container: Vdu_2_Container_1
    - container: Vdu_2_Container_2
    - virtual_storage: opendbDbStorage

Vdu_2_Container_1:
  type: tosca.nodes.nfv.Vdu.OsContainer
  properties:
    name: "opendb FE Container"
    description: "DB Server Container"
    requested_cpu_resources: 1000 # In Milli-Cpus, ie 1 CPU
    cpu_resource_limit: 2500 # In Milli-Cpus, ie 2.5 CPU
    requested_memory_resources: 1 GB
    memory_resource_limit: 10 GB
  artifacts:
    sw_image:
      type: tosca.artifacts.nfv.SwImage
      file: Artifacts/Images/opendbFEimage
properties:
    name: opendb
    version: "4.3"
    checksum:
        algorithm: sha-256
        hash: a411cafee2f0702572369da0b765e2
    container_format: docker
    size: "1024MB"

Vdu_2_Container_2:
    type: tosca.nodes.nfv.Vdu.OsContainer
    properties:
        name: "opendb Container"
        description: "DB Server Container"
        requested_cpu_resources: 1000 # In Milli-Cpus, ie 1 CPU
        cpu_resource_limit: 2500 # In Milli-Cpus, ie 2.5 CPU
        requested_memory_resources: 1 GiB
        memory_resource_limit: 10 GiB
    artifacts:
        sw_image:
            type: tosca.artifacts.nfv.SwImage
            file: Artifacts/Images/opendbBEimage
            properties:
                name: opendb
                version: "4.3"
                checksum:
                    algorithm: sha-256
                    hash: a411cafee2f0702572369da0b765e2
                container_format: docker
                size: "1024MB"
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

All type definitions specified in clause 6 of the present document are contained in the file etsi_nfv_sol001_vnfd_types.yaml of archive gs_nfv-sol001v040201p0.zip which accompanies the present document.

This file is also available at the following URL:


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 “3”, “3” 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

All type definitions specified in clause 7 of the present document are contained in the file etsi_nfv_sol001_nsd_types.yaml of archive gs_nfv-sol001v040201p0.zip which accompanies the present document.

This file is also available at the following URL:


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 "3", "3" 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

All type definitions specified in clause 8 of the present document are contained in the file etsi_nfv_sol001_pnfd_types.yaml of archive gs_nfv-sol001v040201p0.zip which accompanies the present document.

This file is also available at the following URL:

- [https://forge.etsi.org/rep/nfv/SOL001/raw/v4.2.1/etsi_nfv_sol001_pnfd_types.yaml](https://forge.etsi.org/rep/nfv/SOL001/raw/v4.2.1/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 "3", "3" 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 type definitions as specified in clause 9 and used by at least two types of deployment templates are contained in the file etsi_nfv_sol001_common_types.yaml of archive gs_nfv-sol001v040201p0.zip which accompanies the present document.

This file is also available at the following URL:

- [https://forge.etsi.org/rep/nfv/SOL001/raw/v4.2.1/etsi_nfv_sol001_common_types.yaml](https://forge.etsi.org/rep/nfv/SOL001/raw/v4.2.1/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 "3", "3" 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-YAML-v1.3 [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-YAML-v1.3 [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-YAML-v1.3 [20] 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-YAML-v1.3 [20] 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-YAML-v1.3 [20].

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.

| 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   | X    |
| tosca.capabilities.Node    | X    | X   | X    |
| tosca.relationships.Root    | X    |     |      |
| tosca.relationships.DependsOn | X    | X   | X    |
| tosca.interfaces.Root      | X    |     |      |
| tosca.nodes.Root           | X    | X   |      |
| tosca.groups.Root          |     | X   |      |
| tosca.policies.Root        | X    | X   | X    |
| tosca.policiesPlacement   | X    |     |      |
9) When using the TOSCA functions listed in table 5.9-1, it is valid according to section 4 of the TOSCA-Simple-Profile-YAML-v1.3 [20].

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 [22], 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 [22], 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>
<thead>
<tr>
<th>ETSI GS NFV-SOL 001 (the present document)</th>
<th>Type and Property or entity name</th>
<th>Type and attribute name</th>
<th>Data model</th>
<th>Operation (see note 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; descriptor_id</td>
<td>vnfId</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfInfoModificationRequest</td>
<td>Modify VNF Information (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfInfoModifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfLcmOpOcc</td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfLcmOperationOccurrenceNotification</td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfLcmOperationOccurrenceNotification</td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfpkgInfo</td>
<td>Create VNF package (see SOL005) Query/Read VNF Package Info (see SOL003 and SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPackageOnboardingNotification</td>
<td>Notify about VNF Package (see SOL003 and SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPackageChangeNotification</td>
<td>Notify about VNF Package (see SOL003 and SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance</td>
<td>Query NS (see SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UpdateNsRequest</td>
<td>Update NS (see SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsLcmOperationOccurrenceNotification</td>
<td>Notify about NS LCM (see SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PkgmSubscriptionRequest</td>
<td>Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PkgmNotificationsFilter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type and Property or entity name</td>
<td>Data model</td>
<td>Operation (see note 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------</td>
<td>-----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; provider</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Create VNF package (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query/Read VNF Package Info (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; product_name</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Create VNF package (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query/Read VNF Package Info (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; software_version</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Create VNF package (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query/Read VNF Package Info (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NsInstance Query NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgSubscriptionRequest Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VnfPkgInfo Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; descriptor_version</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETSI GS NFV-SOL 001 (the present document)</td>
<td>SOL APIs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type and Property or entity name</strong></td>
<td><strong>Data model</strong></td>
<td><strong>Operation (see note 1)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnfInfoModifications</td>
<td>Modify VNF Information (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnfLcmOpOcc -&gt; VnfInfoModifications</td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnfLcmOperationOccurrenceNotification -&gt; VnfInfoModifications</td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnfPkgInfo</td>
<td>Create VNF package (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query/Read VNF Package Info (see SOL003 and SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstanceId -&gt; VnfInstanceId</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PkgmSubscriptionRequest -&gt; PkgmNotificationsFilter -&gt; vnfProductsFromProviders -&gt; vnfProducts -&gt; versions</td>
<td>Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PkgmSubscription -&gt; PkgmNotificationsFilter -&gt; vnfProductsFromProviders -&gt; vnfProducts -&gt; versions</td>
<td>Subscription about VNF Package (see SOL003 and SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; configurable_properties (KeyValuePairs) vnfConfigurableProperties</td>
<td>VnfInstance</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF -&gt; flavour_id (IdentifierInVnfd) flavourId</td>
<td>VnfInstanceId</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstantiateVnfRequest</td>
<td>Instantiate VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GrantRequest</td>
<td>Grant Lifecycle Operation (see SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstanceId -&gt; VnfInstanceId -&gt; instantiatedVnfInfo</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(IdentifierInVnfd) vnflavourId</td>
<td>UpdateNsRequest -&gt; instantiateVnfData</td>
<td>Update NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(IdentifierInVnfd) newFlavourId</td>
<td>ChangeVnfFlavourRequest</td>
<td>Change VNF Flavour (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.policies.nfv.ScalingAspects -&gt;aspects[IdentifierInVnfd] aspectId</td>
<td>VnfInstance -&gt; instantiatedVnfInfo -&gt; ScaleInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScaleVnfRequest</td>
<td>Scale VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type and Property or entity name</td>
<td>Data model</td>
<td>Operation (see note 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------</td>
<td>-----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScaleVnfToLevelRequest</td>
<td>ScaleInfo</td>
<td>Scale VNF to level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance</td>
<td>VnfInstance</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScaleInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScaleVnfToLevelRequest</td>
<td>ScaleInfo</td>
<td>Scale VNF to level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance</td>
<td>VnfInstance</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScaleInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.datatypes.nfv.ScaleInfo</td>
<td>(integer)</td>
<td>Create VNF Identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-&gt;scale_level</td>
<td>scaleLevel</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Query VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScaleInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnfInstance</td>
<td>VnfInstance</td>
<td>Change VNF Flavour</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Change External VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>Connectivity (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScaleInfo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance</td>
<td>GrantRequest</td>
<td>Grant Lifecycle Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resourceTemplateId</td>
<td>(see SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VnfExtCp</td>
<td>VnfInstance</td>
<td>Create VNF Identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cpdid</td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfExtCpInfo</td>
<td>Query VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Change VNF Flavour</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfExtCpData</td>
<td>Change External VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrantRequest</td>
<td>Connectivity (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfExtCpData</td>
<td>Grant Lifecycle Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrantRequest</td>
<td>(see SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resourceTemplateId</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resourceTemplateId</td>
<td>(see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.VnfVirtualLink</td>
<td>VnfInstance</td>
<td>Create VNF Identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vnfVirtualLinkDescId</td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ExtManagedVirtualLinkInfo</td>
<td>Query VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Change VNF Flavour</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfExtVirtualLinkResourceInfo</td>
<td>Change External VNF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrantRequest</td>
<td>Connectivity (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ExtManagedVirtualLinkData</td>
<td>Grant Lifecycle Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnlCmcOpOcc</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resourceChanges</td>
<td>Get Operation Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AffectedVirtualLink</td>
<td>(see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrantRequest</td>
<td>Grant Lifecycle Operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ExtManagedVirtualLinkData</td>
<td>(see SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ExtManagedVirtualLinkInfo</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfInstance</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>instantiatingVnfInfo</td>
<td>(see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VnfVirtualLinkResourceInfo</td>
<td>Query NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
<td>Data model</td>
<td>Operation (see note 1)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------</td>
<td>------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>IdentifierInVnfd vnfdVirtualLinkDescId</td>
<td>UpdateNsRequest &gt; InstantiateVnfData &gt; ExtManagedVirtualLinkData</td>
<td>Update NS (see SOL005)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IdentifierInVnfd id</td>
<td>Vnfinstance &gt; instantiatedVnfinfo &gt; MonitoringParameter</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance &gt; Vnfinstance &gt; instantiatedVnfinfo &gt; MonitoringParameter</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(String) name</td>
<td>Vnfinstance &gt; instantiatedVnfinfo &gt; MonitoringParameter</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance &gt; Vnfinstance &gt; instantiatedVnfinfo &gt; MonitoringParameter</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(String) localizationLanguage</td>
<td>Vnfinstance &gt; instantiatedVnfinfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance &gt; Vnfinstance &gt; instantiatedVnfinfo</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IdentifierInVnfd vduId</td>
<td>Vnfinstance &gt; instantiatedVnfinfo &gt; VnfcResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnflcmOpOcc &gt; resourceChanges &gt; AffectedVnfc</td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VnflcmOperationOccurrenceNotification &gt; AffectedVnfc</td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NsInstance &gt; Vnfinstance &gt; instantiatedVnfinfo &gt; VnfcResourceInfo</td>
<td>Query NS (see SOL005)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(IdentifierInVnfd) vnfdVirtualComputeDescId</td>
<td>Grant vimAssets &gt; VimComputeResourceFlavour</td>
<td>Grant Lifecycle Operation (see SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IdentifierInVnfd cpdId</td>
<td>Vnfinstance &gt; instantiatedVnfinfo &gt; VnfcResourceInfo &gt; VnfCpInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vnfinstance &gt; instantiatedVnfinfo &gt; VnfcResourceInfo &gt; VnfCpInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003) Query VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstantiateVnRequest &gt; ExtVirtualLinkData &gt; VnfExtCpData</td>
<td>See note 3.</td>
<td>Instantiate VNF (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChangeVnfFlavourRequest &gt; ExtVirtualLinkData &gt; VnfExtCpData</td>
<td>See note 3.</td>
<td>Change VNF Flavour (see SOL002 and SOL003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETSI GS NFV-SOL 001 (the present document)</td>
<td>SOL APIs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type and Property or entity name</strong></td>
<td><strong>Type and attribute name</strong></td>
<td><strong>Data model</strong></td>
<td><strong>Operation (see note 1)</strong></td>
<td></td>
</tr>
<tr>
<td>ChangeExtVnfConnectivityRequest</td>
<td>ExtVirtualLinkData</td>
<td>VnfExtCpData</td>
<td>Change External VNF Connectivity (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NsInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfExtCpInfo</td>
<td>Query NS (see SOL005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) resourceTemplateId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GrantRequest</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResourceDefinition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grant Lifecycle Operation (see SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NsInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Query NS (see SOL005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) resourceTemplateId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GrantRequest</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResourceDefinition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grant Lifecycle Operation (see SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) resourceTemplateId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(IdentifierInVnfd) virtualStorageDescl</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VnfInstanceId</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instantiatedVnfInfo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VirtualStorageResourceInfo</td>
<td>Create VNF Identifier (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Query VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Get Operation Status (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Notify about VNF LCM (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See note 3.</td>
<td></td>
</tr>
<tr>
<td>ETSI GS NFV-SOL 001 (the present document)</td>
<td>SOL APIs</td>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
<td>Data model</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------</td>
<td>---------------------------------</td>
<td>------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>tosca.policies.nfv.InstantiationLevel -&gt; levels[key]</td>
<td></td>
<td>instantiationLevelId</td>
<td>ScaleVnfToLevelRequest</td>
<td>Scale VNF to level (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ChangeVnfFlavourRequest</td>
<td>Change VNF Flavour (see SOL002 and SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GrantRequest</td>
<td>Grant Lifecycle Operation (see SOL003)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UpdateNsRequest -&gt; changeVnfFlavourData</td>
<td>Update NS (see SOL005)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IdentifierInVnfd -&gt; vnfInstantiationLevelId</td>
<td>UpdateNsRequest -&gt; InstantiateVnfData (see note 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>InstantiateVnfRequest</td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters in the inputs of the instantiate operation of the Vnflcm interface See note 2</td>
<td></td>
<td>(KeyValuePairs) additionalParams</td>
<td>InstantiateVnfRequest</td>
<td>Instantiate VNF (see SOL002 and SOL003)</td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters in the inputs of the scale operation of the Vnflcm interface See note 2</td>
<td></td>
<td>(KeyValuePairs) additionalParams</td>
<td>ScaleVnfRequest</td>
<td>Scale VNF (see SOL002 and SOL003)</td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters in the inputs of the 'scale to level' operation of the Vnflcm interface See note 2</td>
<td></td>
<td>(KeyValuePairs) additionalParams</td>
<td>ScaleVnfToLevelRequest</td>
<td>Scale VNF to level (see SOL002 and SOL003)</td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters in the inputs of the 'change vnf flavour operation' of the Vnflcm interface See note 2</td>
<td></td>
<td>(KeyValuePairs) additionalParams</td>
<td>ChangeVnfFlavourRequest</td>
<td>Change VNF Flavour (see SOL002 and SOL003)</td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.VnfOperationAdditionalParameters in the inputs of the operate operation of the Vnflcm interface See note 2</td>
<td></td>
<td>(KeyValuePairs) additionalParams</td>
<td>OperateVnfRequest</td>
<td>Operate VNF (see SOL002 and SOL003)</td>
</tr>
<tr>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
<td>Data model</td>
<td>Operation (see note 1)</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------</td>
<td>------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the heal operation of the Vnflcm interface See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>HealVnfRequest</td>
<td>Heal VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the 'change external vnf connectivity' operation of the Vnflcm interface See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>ChangeExtVnfConnectivityRequest</td>
<td>Change External VNF Connectivity (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the terminate operation of the Vnflcm interface See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>TerminateVnfRequest</td>
<td>Terminate VNF (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the create snapshot operation of the Vnflcm interface See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>CreateVnfSnapshotRequest</td>
<td>CreateVNFSnapshot (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the revert to snapshot operation of the Vnflcm interface See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>RevertToVnfSnapshotRequest</td>
<td>RevertToVNFSnapshot (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>Properties of <code>tosca.datatypes.nfv.VnfOperationAdditionalParameters</code> in the inputs of the Change current VNF operation of the Vnflcm or ChangeCurrentVnfPackage interface. See note 2</td>
<td>(KeyValuePairPairs) additionalParams</td>
<td>ChangeCurrentVnfPkgRequest</td>
<td>Change current VNF package (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>name of type <code>tosca.nodes.nfv.Vdu.Compute</code> Node template name of type <code>tosca.nodes.nfv.Vdu.VirtualBlockStorage</code></td>
<td>(IdentifierInVnfd) id</td>
<td>VnfPkgInfo -&gt; VnfPackageSoftwareImageInfo</td>
<td>Create VNF Package Info (see SOL005)</td>
<td>Query VNF Package Info (see SOL005)</td>
</tr>
<tr>
<td>Name of attribute for VNF indicator in VNF node</td>
<td>(IdentifierInVnfd) id</td>
<td>VnfIdIndicator</td>
<td>Get indicator value (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td><strong>ETSI GS NFV-SOL 001</strong> (the present document)</td>
<td><strong>SOL APIs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type and Property or entity name</strong></td>
<td><strong>Type and attribute name</strong></td>
<td><strong>Data model</strong></td>
<td><strong>Operation (see note 1)</strong></td>
<td></td>
</tr>
<tr>
<td>Name of attribute for VNF indicator in VNF</td>
<td>(String) Name</td>
<td>VnfIndicatorValueChangeNotification</td>
<td>Notify about VNF indicator value change (see SOL002 and SOL003).</td>
<td></td>
</tr>
<tr>
<td>Value of attribute for VNF indicator in VNF node</td>
<td>(Object) value</td>
<td>VnfIndicatorValueChangeNotification</td>
<td>Notify about VNF indicator value change (see SOL002 and SOL003).</td>
<td></td>
</tr>
<tr>
<td>Node template name of type tosca.nodes.nfv.Vdu.Compute</td>
<td>(IdentifierInVnfd) vnfdSoftwareImageId</td>
<td>Grant</td>
<td>Grant Lifecycle Operation (see SOL003)</td>
<td></td>
</tr>
<tr>
<td>Node template name of type tosca.nodes.nfv.Vdu.VirtualBlockStorage</td>
<td>(Identifier) vnfdId</td>
<td>ChangeCurrentVnfPkgRequest</td>
<td>Change current VNF package (see SOL002 and SOL003)</td>
<td></td>
</tr>
<tr>
<td>tosca.datatypes.nfv.VnfPackageChangeSelector</td>
<td>coordinationActionName</td>
<td>LcmCoordRequest</td>
<td>CoordinateLcmOperation for VNF LCM (see SOL002)</td>
<td></td>
</tr>
<tr>
<td>tosca.policies.nfv.LcmCoordinationsForLcmOperation</td>
<td>coordinationActionName</td>
<td>LcmCoordRequest</td>
<td>CoordinateLcmOperation for Change current VNF package (see SOL002)</td>
<td></td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.InputOpCoordParams</td>
<td>inputParameters</td>
<td>LcmCoordRequest</td>
<td>CoordinateLcmOperation for VNF LCM and Change current VNF package (see SOL002)</td>
<td></td>
</tr>
<tr>
<td>Properties of tosca.datatypes.nfv.OutputOpCoordParams</td>
<td>outputParameters</td>
<td>LcmCoord</td>
<td>CoordinateLcmOperation for VNF LCM and Change current VNF package (see SOL002)</td>
<td></td>
</tr>
<tr>
<td>(KeyValuePairs) VnfInfoModificationRequest</td>
<td>Modify VNF Information</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>ETSI GS NFV-SOL001</th>
<th>SOL APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS -&gt; descriptor_id</td>
<td>(Identifier) nsdId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(Identifier) nsdId</td>
<td>NsdInfo</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NsdOnboardingNotification</td>
</tr>
<tr>
<td></td>
<td>NsdOnboardingFailureNotification</td>
</tr>
<tr>
<td></td>
<td>NsdChangeNotification</td>
</tr>
<tr>
<td></td>
<td>NsdDeletionNotification</td>
</tr>
<tr>
<td></td>
<td>NsdmSubscriptionRequest</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NsdmSubscription</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateNsRequest</td>
<td>Create NS</td>
</tr>
<tr>
<td></td>
<td>NsInstanceId</td>
</tr>
<tr>
<td></td>
<td>Query NS</td>
</tr>
<tr>
<td></td>
<td>Delete NS</td>
</tr>
<tr>
<td></td>
<td>NsLcmOpOcc</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NsLcmOperationOccurrenceNotification</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(Identifier) newNsdId</td>
<td>UpdateNsRequest</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS -&gt; name</td>
<td>(String) nsdName</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NsdmSubscriptionRequest</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ETSI GS NFV-SOL001</td>
<td>SOL APIs</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>Type and Property or entity name</strong></td>
<td><strong>Type and attribute name</strong></td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS</td>
<td>(Version) nsdVersion</td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS</td>
<td>(String) nsdDesigner</td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS</td>
<td>(Identifier) nsdInvariantId</td>
</tr>
<tr>
<td>tosca.nodes.nfv.NS</td>
<td>(Identifier) nsFlavourId</td>
</tr>
<tr>
<td>tosca.nodes.nfv.VNF</td>
<td>(Identifier) vProfileId</td>
</tr>
<tr>
<td>Node template name in NSD of type derived from tosca.nodes.nfv.NS</td>
<td>(Identifier) nsProfileId</td>
</tr>
<tr>
<td>Node template name in NSD of type derived from tosca.nodes.nfv.VNF</td>
<td>(Identifier) vProfileId</td>
</tr>
<tr>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>NsLcmOperationOccurrenceNotification</td>
<td>-&gt; AffectedVnf</td>
</tr>
<tr>
<td>NsInstance</td>
<td>-&gt; NsAffinityOrAntiAffinityRule</td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; NsAffinityOrAntiAffinityRule</td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; ParamsForVnf</td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; VnfLocationConstraint</td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; VnfInstanceData</td>
</tr>
<tr>
<td>ScaleNsRequest</td>
<td>-&gt; ScaleNsData</td>
</tr>
<tr>
<td>ScaleNsRequest</td>
<td>-&gt; VnfLocationConstraint</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td>-&gt; VnfInstanceData</td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(Identifier) pnfId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(Identifier) pnfId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>NsLcmOpOcc</td>
<td>-&gt; AffectedPnf</td>
</tr>
<tr>
<td>NsLcmOperationOccurrenceNotification</td>
<td>-&gt; AffectedPnf</td>
</tr>
<tr>
<td>NsInstance</td>
<td>-&gt; PnfInfo</td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; AddPnfData</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td>-&gt; AddPnfData</td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(String) pnfdName</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(Version) pnfdVersion</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(String) pnfdProvider</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tosca.nodes.nfv.PNF</td>
<td>(Identifier) pnfdInvariantId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Node template name in NSD of type derived from tosca.nodes.nfv.PNF

<p>| (IdentifierInNsD) pnfProfileId | NsLcmOpOcc | Query/read information about NS LCM (see SOL005) |
| | -&gt; AffectedPnf | |
| NsLcmOperationOccurrenceNotification | -&gt; AffectedPnf | Notification about NS LCM (see SOL005) |</p>
<table>
<thead>
<tr>
<th>ETSI GS NFV-SOL001</th>
<th>SOL APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type and Property or entity name</td>
<td>Type and attribute name</td>
</tr>
<tr>
<td>NsInstance</td>
<td>-&gt; PnfdInfo</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>-&gt; AddPnfdData</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td>-&gt; AddPnfdData</td>
</tr>
<tr>
<td>Node template name of type tosca.nodes.nfv.Cp</td>
<td>(IdentifierInNs) cpdId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>Create NS</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td>Delete NS</td>
</tr>
<tr>
<td>Group template name of type tosca.groups.nfv.VNFFG</td>
<td>(IdentifierInNs) vnffgId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>InstantiateNsRequest</td>
<td>Instantiate NS</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td>Update NS</td>
</tr>
<tr>
<td>UpdateNsRequest</td>
<td></td>
</tr>
<tr>
<td>Group template name of type tosca.groups.nfv.VNFFG</td>
<td>(IdentifierInNs) vnffgId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Group template name of type tosca.groups.nfv.VNFFG</td>
<td>(IdentifierInNs) vnffgId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>See note 2</td>
<td>(IdentifierInNs) aspectId</td>
</tr>
<tr>
<td>See note 2</td>
<td>(IdentifierInNs) nsScalingAspectId</td>
</tr>
<tr>
<td>See note 2</td>
<td>(Identifier) vnfPkgIds</td>
</tr>
<tr>
<td>See note 2</td>
<td>(IdentifierInNs) healScript</td>
</tr>
<tr>
<td>See note 2</td>
<td>(IdentifierInNs) aspectId</td>
</tr>
<tr>
<td>See note 2</td>
<td>(IdentifierInNs) nsScalingAspectId</td>
</tr>
<tr>
<td>ETSI GS NFV-SOL001</td>
<td>Type and Property or entity name</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Node template name of type tosca.nodes.nfv.Sap</td>
<td>(IdentifierInNsd) sapId</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>InstantiateNsRequest</td>
</tr>
<tr>
<td></td>
<td>UpdateNsRequest</td>
</tr>
<tr>
<td></td>
<td>NsLcmOpOcc</td>
</tr>
<tr>
<td></td>
<td>NsLcmOperationOccurrenceNotification</td>
</tr>
</tbody>
</table>

**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 versions 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 fulfill the NS LCM operations described in ETSI GS NFV-IFA 013 [1.8]. 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 NS 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:

- \(<\text{NS}_\text{LCM}\_\text{base}\_\text{operation}\_\text{workflow}\_\text{name}>\_\text{start}_\text{<step}\_\text{name}>\) for preamble steps
- \(<\text{NS}_\text{LCM}\_\text{base}\_\text{operation}\_\text{workflow}\_\text{name}>\_\text{end}_\text{<step}\_\text{name}>\) 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. \(<\text{NS}_\text{LCM}\_\text{base}\_\text{operation}\_\text{workflow}\_\text{name}>\)
- 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:

```yaml
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
  - example_vnf1.yaml
  - example_vnf2.yaml
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:
        operations:
          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:
#    operations:
#      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:
#    operations:
#      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: MyCompany.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:
    #.
    target: my_service
    activities: []
    on_success:
      - create_VNF_1

  create_VNF_1:  # Step: Instantiate VNF_1
    target: VNF_1
    activities:
- call_operation: 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:
  - create_VNF_2
# . . .
create_VNF_2: # Step: Instantiate VNF_2
target: VNF_2
activities:
  - call_operation: 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:
# . . .
target: my_service
activities: []
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:
# . . .
target: my_service
activities: []
on_success:
  - terminate_VNF_2
# steps for terminate workflow
terminate_VNF_1: # Step: Terminate VNF_1
target: VNF_1
activities:
  - call_operation: 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_VNF_1
# . . .
terminate_VNF_2: # Step: Terminate VNF_2
target: VNF_2
activities:
  - call_operation: 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:
  target: my_service
  activities: []
  
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 GS

F.1 Introduction

This annex provides the list of non-backward compatible changes during the development of the present document.

A change introduced in version n of the present document is non-backward compatible if a service template written according to a previous version n-1 of the present document, i.e. a service template that has not been updated according to this change, is invalid with respect to this change for a NFVO/VNFM compliant to version n of the present document.

This annex focuses on compatibility from a descriptor viewpoint. It does not evaluate whether a change made to the present document leads to non-backward compatible changes on the APIs referenced in Annex D 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 2.6.1 [i.22] and version 2.7.1 [i.23] of the present document.

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>nfvi_constraints type changed from list of string to map of string.</td>
<td>6.8.3</td>
</tr>
<tr>
<td>2</td>
<td>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.</td>
<td>6.10.13</td>
</tr>
<tr>
<td>3</td>
<td>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.</td>
<td>6.9</td>
</tr>
<tr>
<td>4</td>
<td>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.</td>
<td>6.10.10</td>
</tr>
<tr>
<td>5</td>
<td>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.</td>
<td>6.10.13</td>
</tr>
<tr>
<td>6</td>
<td>boot_order type changed from list of string to Boolean.</td>
<td>6.8.3</td>
</tr>
<tr>
<td>7</td>
<td>boot_data type changed from string to BootData data type.</td>
<td>6.8.3</td>
</tr>
</tbody>
</table>

F.3 Non-Backward Compatible changes between version 2.8.1 and 3.3.1

Table F.3-1 provides a list of non-backward compatible changes between version 2.8.1 [i.20] and version 3.3.1 [i.19] of the present document.
Table F.3-1: Non-backward compatible changes

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Multiple VNF deployment flavour design changed from using TOSCA v 1.2 grammar to TOSCA v 1.3 grammar:</td>
<td>6.11.2</td>
</tr>
<tr>
<td></td>
<td>- in the top level service templates, removing the imports for lower level service templates. See note;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- in the low level service templates, use of substitution filter instead of property mapping.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Multiple NS deployment flavour design changed from using TOSCA v 1.2 grammar to TOSCA v 1.3 grammar:</td>
<td>7.11.2</td>
</tr>
<tr>
<td></td>
<td>- in the top level service templates, removing the imports for lower level service templates. See note;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- in the low level service templates, use of substitution filter instead of property mapping.</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Instead the low level service templates are declared in the Other-Definitions of the TOSCA.meta file as specified in TOSCA-Simple-Profile-YAML-v1.3 [20].

F.4 Non-Backward Compatible changes between version 3.3.1 and 3.5.1

Table F.4-1 provides a list of non-backward compatible changes between version 3.3.1 [i.19] and version 3.5.1 [i.21] of the present document.

Table F.4-1: Non-backward compatible changes

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In VNF, Vdu.Compute and VnfVirtualLink node type definition, the type of monitoring_parameters changed from list to map.</td>
<td>6.8.1, 6.8.3, 6.8.9</td>
</tr>
<tr>
<td>2</td>
<td>In NsMonitoring policy definition, the type of ns_monitoring_parameters changed from list to map.</td>
<td>7.10.4</td>
</tr>
<tr>
<td>3</td>
<td>In VnfMonitoring policy definition, the type of vnf_monitoring_parameters changed from list to map.</td>
<td>7.10.5</td>
</tr>
</tbody>
</table>
Annex G (informative):
Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Information about changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016.05</td>
<td>0.0.1</td>
<td>Implemented NFVSOL(16)000005r1, GS_NFV_SOL001_ToC</td>
</tr>
<tr>
<td>2016.07</td>
<td>0.02</td>
<td>Implemented NFVSOL(16)000026r1, NFVSOL(16)000027r1, NFVSOL(16)000028r2</td>
</tr>
<tr>
<td>2017.09</td>
<td>0.1.0</td>
<td>Implemented NFVSOL(17)000539r3, NFVSOL(17)000540r3, NFVSOL(17)000542r2, NFVSOL(17)000544r1</td>
</tr>
<tr>
<td>2017.10</td>
<td>0.2.0</td>
<td>Implemented NFVSOL(17)000545r4, NFVSOL(17)000559r2, NFVSOL(17)000560r2, NFVSOL(17)000570r1, NFVSOL(17)000575r1, NFVSOL(17)000616r1, NFVSOL(17)000641</td>
</tr>
<tr>
<td>2017.11</td>
<td>0.3.0</td>
<td>Clean-up done by editHelp</td>
</tr>
<tr>
<td>2017.12</td>
<td>0.4.0</td>
<td>Implemented NFVSOL(17)000642r1, NFVSOL(17)000675r1, NFVSOL(17)000756</td>
</tr>
<tr>
<td>2018.01</td>
<td>0.5.0</td>
<td>Implemented NFVSOL(17)000621r6, NFVSOL(17)000676r3, NFVSOL(17)000677r2, NFVSOL(17)000736r1, NFVSOL(18)000004r2</td>
</tr>
<tr>
<td>2018.03</td>
<td>0.6.0</td>
<td>Implemented NFVSOL(18)000049, NFVSOL(18)000052r1, NFVSOL(18)000048R1, NFVSOL(18)000040, NFVSOL(18)000023r2, NFVSOL(18)000024r2, NFVSOL(18)000029r1, NFVSOL(18)000063r5, NFVSOL(18)000077, NFVSOL(18)000046r2, NFVSOL(18)000074r2, NFVSOL(18)000075r1, NFVSOL(18)0000611r5, NFVSOL(18)000041r1, NFVSOL(18)000028, NFVSOL(18)000044r2, NFVSOL(18)000055r2, NFVSOL(18)000045r1</td>
</tr>
<tr>
<td>2018.03</td>
<td>0.6.1</td>
<td>Implemented NFVSOL(18)000052r1, NFVSOL(18)000009r4, NFVSOL(18)000043r3, NFVSOL(18)0000112r1, NFVSOL(18)0000117r2, NFVSOL(18)0000124r1, NFVSOL(18)0000129r1, NFVSOL(18)0000115r2</td>
</tr>
<tr>
<td>2018.04</td>
<td>0.6.2</td>
<td>Implemented NFVSOL(18)000113r2, NFVSOL(18)000119r2, NFVSOL(18)000121, NFVSOL(18)000135, NFVSOL(18)000116r1, NFVSOL(18)000157, NFVSOL(18)000125r1, NFVSOL(18)000158r1</td>
</tr>
<tr>
<td>2018.05</td>
<td>0.6.3</td>
<td>Implemented NFVSOL(18)000168, NFVSOL(18)000169r2, NFVSOL(18)000173, NFVSOL(18)000174r2</td>
</tr>
<tr>
<td>2018.05</td>
<td>0.7.0</td>
<td>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</td>
</tr>
<tr>
<td>2018.06</td>
<td>0.8.0</td>
<td>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</td>
</tr>
<tr>
<td>2018.06</td>
<td>0.9.0</td>
<td>Implemented NFVSOL(18)000289r2, Adding etsi_nfv_sol001_vnfd_0_9_0_type.yaml and SOL001 Graphics v0_9_0.pptx in the draft GS zip package</td>
</tr>
<tr>
<td>2018.08</td>
<td>0.10.0</td>
<td>Adding editorial changes for all the TOSCA type definitions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add Editorial modification for clause numbering and format

Adding etsi_nfv_sol001_vnfd_0_9_0_type.yaml and SOL001 Graphics v0_9_0.pptx in the draft GS zip package

Editors changes for all the TOSCA type definitions
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Information about changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018.09</td>
<td>0.11.0</td>
<td>Implemented NFVSOL(18)000486r7, NFVSOL(18)000495r1, NFVSOL(18)000497r3, NFVSOL(18)000498, NFVSOL(18)000500r3, NFVSOL(18)000503r1, 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</td>
</tr>
<tr>
<td>2018.10</td>
<td>0.12.0</td>
<td>Implemented NFVSOL(18)000507r3, NFVSOL(18)000621r4, NFVSOL(18)000547r1, NFVSOL(18)000567r1, NFVSOL(18)000574, NFVSOL(18)000579, NFVSOL(18)000590r1</td>
</tr>
<tr>
<td>2018.11</td>
<td>0.13.0</td>
<td>Implemented NFVSOL(18)000563r5, NFVSOL(18)000569r5, NFVSOL(18)000575r5, NFVSOL(18)000586r2, NFVSOL(18)000587r1, NFVSOL(18)000589r2, NFVSOL(18)000591, NFVSOL(18)000604r2, NFVSOL(18)000606r1, NFVSOL(18)000607r5, NFVSOL(18)000614r1, 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)000653r2, NFVSOL(18)000658r3, NFVSOL(18)0006659, NFVSOL(18)000660, NFVSOL(18)000665r1, NFVSOL(18)000666, NFVSOL(18)000678r2, NFVSOL(18)000679, NFVSOL(18)000682, NFVSOL(18)000684r1, NFVSOL(18)000687</td>
</tr>
<tr>
<td>2019.03</td>
<td>2.5.2</td>
<td>Implemented NFVSOL(19)000063r4, NFVSOL(19)000068r1, NFVSOL(19)000069r2, NFVSOL(19)000070r1, NFVSOL(19)000080, NFVSOL(19)00107 r1, NFVSOL(19)00120r1, NFVSOL(19)000121, NFVSOL(19)00039r5, NFVSOL(19)00087r2, NFVSOL(19)000675, NFVSOL(19)00077, NFVSOL(19)00082r3, NFVSOL(19)00085, NFVSOL(19)00086, NFVSOL(19)000106r1</td>
</tr>
<tr>
<td>2019.03</td>
<td>2.5.3</td>
<td>Implemented NFVSOL(19)000084r4, NFVSOL(19)000101r2, NFVSOL(19)000165, NFVSOL(19)000166, NFVSOL(19)000167r1, NFVSOL(19)000170r4, NFVSOL(19)000173, NFVSOL(19)000163, NFVSOL(19)000119</td>
</tr>
</tbody>
</table>
| 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 |
<p>| 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)000268r7 |
| 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 |</p>
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Information about changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.11</td>
<td>2.6.6</td>
<td>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_clause_9_1, NFVSOL(19)000595_SOL001ed271 resolving_requirement_occurrence_issue, NFVSOL(19)000597r3_SOL001ed271 deployment_flavour_related_CSAR_design_for_NSD, NFVSOL(19)000599r2_SOL001ed271 TOSCA_Imperative_workflows_NSD_Editor_s_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)000617r3_SOL001ed271 nfviConstraint, NFVSOL(19)000633_SOL001ed271 VnfIndicators_editor_s_notes_resolution, NFVSOL(19)000636r2_SOL001ed271 NS_workflow_definition_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_VNF, 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</td>
</tr>
<tr>
<td>2019.11</td>
<td>2.6.7</td>
<td>Implemented NFVSOL(19)000710r3_SOL001ed331 deployment_flavour_related_CSAR_design_for_VNF, NFVSOL(19)000792r1_SOL001ed331_FEAT10_Add_specification_for_Multi-Site_Connectivity_Services, NFVSOL(19)000799r2_SOL001ed331_FEAT05_Adding_priority_to_NSD</td>
</tr>
<tr>
<td>2020.02</td>
<td>3.0.1</td>
<td>Implemented: NFVSOL(19)000714r4_SOL001ed271 support_of_auto-scaling_with_use_of_VNF_indicator, NFVSOL(20)000018r2_SOL001ed331 deployment_flavour_related_CSAR_design_for_NSD, NFVSOL(20)000067r1_SOL001ed331 FEAT16 Adding_SAL_to_NSD</td>
</tr>
<tr>
<td>2020.02</td>
<td>3.0.2</td>
<td>Implemented: NFVSOL(19)000406r6_SOL001ed331 Criteria_for_backward_compatibility_of_changes, NFVSOL(20)000117_SOL001ed331 Vnfcm_update_TOSCA_1_3_grammar, NFVSOL(20)000172r1_SOL001ed331 VNF_specific_datatypes Naming_rules, NFVSOL(20)000239r2_SOL001ed331 release_3_mirror_adding_NS_DF_design_principle_in_annex, NFVSOL(20)000242_SOL001ed331 updating_annex_C, NFVSOL(20)000285_SOL001ed331 VNF_specific_types Naming_rules, rapporteur changes: editorial, changing &quot;will&quot; to &quot;with&quot; in some places in annex C.2 and C.3, editorial, removing extra space in some of the TOSCA type definitions</td>
</tr>
<tr>
<td>2020.04</td>
<td>3.0.3</td>
<td>Implemented: NFVSOL(20)000016r6_SOL001ed331_FEAT02_VnfPackageChange, NFVSOL(20)000040r6_SOL001ed331_Criteria_for_backward_compatibility_of_changes, NFVSOL(20)000117_SOL001ed331 Vnfcm_update_TOSCA_1_3_grammar, NFVSOL(20)000172r1_SOL001ed331 VNF_specific_datatypes Naming_rules, NFVSOL(20)000239r2_SOL001ed331 release_3_mirror_adding_NS_DF_design_principle_in_annex, NFVSOL(20)000242_SOL001ed331 updating_annex_C, NFVSOL(20)000285_SOL001ed331 VNF_specific_types Naming_rules, rapporteur changes: editorial, changing &quot;will&quot; to &quot;with&quot; in some places in annex C.2 and C.3, editorial, removing extra space in some of the TOSCA type definitions</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Information about changes</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2020.05</td>
<td>3.0.4</td>
<td>Implemented: NFVSOL(20)000295r3_SOL001ed331_FEAT02_ChangeCurrentVnfPackage_interface, NFVSOL(20)000320r6_SOL001ed331_FEAT15_VNF_Snapshot, NFVSOL(20)000354r1_SOL001ed331_FEAT02_ChangeCurrentVnfPackage_AnnexA_D_mapping, NFVSOL(20)000327r1_SOL001ed331_rel3_mirror_corrections_of_specific_node_type, NFVSOL(20)000353_SOL001ed331_Vnfcm_interface_EN_cleanup, NFVSOL(20)000376r2_SOL001ed331_mirror_of_375_PNFDF_geographic_coordinatesスーパー, NFVSOL(20)000378r2_SOL001ed331_rel3_mirror_add_VNF_related_type_names_in_clause_7, NFVSOL(20)000382r1_SOL001ed331_rel3_mirror_clarification_of_using_VnfMonitoring, NFVSOL(20)000384_SOL001ed331_rel3_mirror_clarification_on_ip_address_type, NFVSOL(20)000386_SOL001ed331_rel3_mirror_clarification_on_VirtualLinkable_cap, NFVSOL(20)000194r3_SOL001ed331_release_3_mirror_example_corrections</td>
</tr>
<tr>
<td>2020.06</td>
<td>3.0.5</td>
<td>Implemented: NFVSOL(20)000267r3_SOL001ed331_VNF_node_type_definitions, NFVSOL(20)000355r2_SOL001ed331_examples_cleanup_tosca1_3_interfaces, NFVSOL(20)000356_SOL001ed331_example_cleanup_to_tosca_1_3, NFVSOL(20)000357_SOL001ed331_all_type_definitions_metadata_cleanup, NFVSOL(20)000394r2_SOL001ed331_clause6_11_cleanup_tosca1_3, NFVSOL(20)000395r4_SOL001ed331_sw_image_data_align_with_TOSCA_1_3, NFVSOL(20)000396r5_SOL001ed331_removing_reference_of_TOSCA_1_1_and_1_2, NFVSOL(20)000397r1_SOL001ed331_clause_7_11_align_tosca_1_3, NFVSOL(20)000399r1_SOL001ed331_clause_8_11_align_tosca_1_3, NFVSOL(20)000517r1_SOL001ed331_Rel_3_mirror-_fixing_optional_properties, NFVSOL(20)000527r4_SOL001ed331_update_annex_F, NFVSOL(20)000597_SOL001ed331_resolve_remaining_issues_for_TOSCA_reference, NFVSOL(20)000598r1_SOL001ed331_resolve_editor_note_for_package_change, Undo the implementation of NFVSOL(19)000799r2, supporting for FEAT5 will be removed from this version</td>
</tr>
<tr>
<td>2020.09</td>
<td>3.3.2</td>
<td>Implemented: NFVSOL(20)000387r3_SOL001ed341_adding_virtualLinkProtocolData_for_NsVirtualLink, NFVSOL(20)000617r7_SOL001ed341_Use_of_Credentials_datatype, NFVSOL(20)000618_SOL001ed341_VnfPackageChange_corrections, NFVSOL(20)000674_SOL001ed341_modifiable_attributes_example_correction, NFVSOL(20)000676r1_SOL001ed341_update_extension_rule</td>
</tr>
<tr>
<td>2020.11</td>
<td>3.3.3</td>
<td>Implemented: NFVSOL(20)000723r4_SOL001Ed341_Fixing_examples, NFVSOL(20)000359r5_SOL001ed331_support_of_trunk_port_topology, NFVSOL(20)000679r2_SOL001ed341_monitoring-parameter_identifier, NFVSOL(20)000716r2_SOL001Ed341_Clarification_on_node_type_definitions_in_an_NSD, NFVSOL(20)000737r1_SOL001Ed341_Ambiguous_use_of_may_not, NFVSOL(20)000744r2_SOL001ed341_uniform_delta_correction, NFVSOL(20)000747r1_SOL001ed351_fix_TOSCA_YAML_version_reference, NFVSOL(20)000749r1_SOL001ed341_Adding_NS_scaling_aspects_instantiation_levels, NFVSOL(20)000752r6_SOL001ed351_support_using_VnfConfigurableProperties_for_boot_data, NFVSOL(20)000767_SOL001ed351_Removal_of_constraint_in_VnfIndicator_attribute</td>
</tr>
<tr>
<td>2020.12</td>
<td>3.3.4</td>
<td>Implemented: NFVSOL(20)000774r1_SOL001ed351_add_Dependencies_in_VNFD, NFVSOL(20)000775_SOL001ed351_adding_example_in_NSD_for_VL_protocol_data, NFVSOL(20)000776_SOL001ed351_fix_issues_for_VnfcConfigurableProperties, NFVSOL(20)000777_SOL001_Correct_typo_in_NsAffinityRules_etc, NFVSOL(20)000798_SOL001ed351_Corrections_in_policies_definitions</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Information about changes</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2021.02</td>
<td>3.3.5</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000001_SOL001ed351__Annex_A__-Fixing_errors_in_YAML_examples_A1-A6,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000003r1_SOL001ed351__Annex_A__-Fixing_errors_in_YAML_examples_A7-A17,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000004r1_SOL001ed351__Use_of_TOSCA_functions__-specification,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000007_SOL001ed351_Scale_Inputs,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000008_SOL001ed351_Wrongindentation_of_entry_schema,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000010_SOL001ed351_vducp_occurrences_correction,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000089_SOL001ed351_virtual_binding_capability,</td>
</tr>
<tr>
<td>2021.02</td>
<td>4.0.1</td>
<td>Created based on SOL001 V3.3.5</td>
</tr>
<tr>
<td>2021.03</td>
<td>4.0.2</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000011r8_SOL001ed421_Extensions_to_VNFD_Data_Model_in_Support_of_Container,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000018r6_SOL001ed421_OsContainer_and_OsContainerGroup_node_definition,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000020r2_SOL001ed421_Changes_to_tosca_relationships_nfv_AttachesTo_description_in_support_of_OsContainers,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000064r5_SOL001ed421_Mciop_Profile_modelling,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000118_SOL001ed421_Mirror_SOL118_Adding_VnfLcmCoordination_to_interfacename,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000019r4_SOL001ed421_Enhancements_to_VduCp_in_support_of_containers,</td>
</tr>
<tr>
<td>2021.04</td>
<td>4.0.3</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000021r4_SOL001ed421_New_Nodes__Capabilities__Relationships_and_Datatypes_to_model_VirtualCp,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000171_SOL001ed421_release_4_mirror_adding_VnffIndicator_support_in_NSDF,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000172_SOL001ed421_rel_4_mirror_add_new_NsAutoScale_policy,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000173_SOL001ed421_rel_4_mirror_new_NsVnfIndicator_interface,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000174_SOL001ed421_rel_4_mirror_NS_node_attribute_for_VnfIndicator,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000193_SOL001ed421_Add_per_vnf_instance_property_to_Storage_Nodes,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000213_SOL001ed421_rel4_mirror_add_new_input_in_Nslcm_operation_to_support_autoscale,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000219_SOL001ed421_Rel-4_mirror_of_131__VipCp_node_property,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000220_SOL001ed421_Rel-4_mirror_of_159__FEAT05_Adding_priority_to_NSDF,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000221_SOL001ed421__Corrections_to_Annex_A_examples__mirror_of_120,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000223_SOL001ed421_Change_name_of_container_deployable_unit_requirement,</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Information about changes</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2021.04</td>
<td>4.0.4</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000211r3_SOL001ed421_VNFD_TOSCA_model_update,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000214_SOL001ed421_remove_sw_image_data,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000227_SOL001ed421__VIP_CP_requirements__Mirror_of_122r1__</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000228_SOL001ed421__Support_of_additional_TOSCA_Functions__Mirror_of_121r4,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000229_SOL001ed421_Rel-4_mirror_of_212r1_correct_valid_values_fora,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000238r2_SOL001ed421_Additional_VduCp_vnic_types_in_support_of_containers,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000246r1_SOL001ed421_Correct_some_minor_errors_in_the_definitions_of_Mciop_and_OsContainerDeployableUnit,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000254_SOL001ed421_rel4_mirror_correct_NSD_VNFFG_example,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000256_SOL001ed421__Support_of_additional_TOSCA_Intrinsic_Functions,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000259_SOL001ed421_rel4_mirror_update_table_in_annex_A_9,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000260r1_SOL001ed421_rel4_mirror_multiple_errors_correction,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000262_SOL001ed421_Rel-4_mirror_of_208_Resolution_of_editors_notes,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000263_SOL001ed421_Rel-4_mirror_of_234_Resolution_of_editor_s_note,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000266_SOL001ed421_rel4_mirror_clarification_on_virtualinkProtocol,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000267_SOL001ed421_mirror_of_060r7_VnfLcmOperationCoordination,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000268_SOL001ed421_mirror_of_061_VnfLcmOperationCoordination_extensions,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000269_SOL001ed421_mirror_of_062_VnfLcmOperationCoordination_extensions,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000298_SOL001ed421__Inconsistencies_in_VNF_and_NS_Node_templates__.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000300_SOL001ed421__properties_and_requirements_in_top-level_template,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000302_SOL001ed421__Use_of_the_substitution_directive_in_top-level</td>
</tr>
<tr>
<td>2021.06</td>
<td>4.0.5</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000152r5_SOL001ed421_Helm_chart_artifact_type,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000311_SOL001Ed421_Small_fixes_in_Annex_E,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000312r2_SOL001Ed421_Small_fixes_in_Annex_A,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000313r1_SOL001Ed421_Small_fixes_in_Yaml_definitions,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000336_SOL001Ed351_Clariﬁcation_on_the_contents_of_service_template,</td>
</tr>
<tr>
<td>2021.07</td>
<td>4.0.6</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000364r3_SOL001Ed421_Fixing_issues_in_Annex_A,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000370_SOL001Ed421_Fixing_issues_in_Annex_E,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000373r1_SOL001ed421_update_Swimage_artifact_definition,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000374_SOL001ed421_adding_affinityOrAntiAffinity_scope_for_CIS_node,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000377_SOL001ed421_New_scope_value_for_Ns_affinity_policies,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000395_SOL001ed421_Per_instance_storage,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000409_SOL001ed421_Mirror_408_VnfPackageChange_add_targets,</td>
</tr>
<tr>
<td>2021.09</td>
<td>4.0.7</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000252r3_SOL001ed421_Container_requests_of_extended_resources,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000369r3_SOL001ed421_Fixing_issues_in_Annex_A_14__VNFFG__,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000375r3_SOL001ed421_input_example_for_VnfCOnfigurableProperties,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000407_SOL001ed421_Mirror_of_406_Bugfix_with_adding_fixedIpAddresses,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000435r1_SOL001ed421_swimage_for_virtualBlockStorage,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000436r2_SOL001ed421_updating_placementGroup_for_container,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000450_SOL001ed421_Missing_note_in_Swimage_artifact,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000451r1_SOL001ed421_mcio_constraint_parameters,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000437r2_SOL001ed421_VnfExtCp_updating,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000480_SOL001ed421_mcio_artifact_clariﬁcation,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000481r1_SOL001ed421_SecurityGroupRule_for_container,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000484_SOL001ed421_instantiation_levels_and_scaling_aspects,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000487_SOL001ed421_resolution_of_Helm_charts_related_editors_notes,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000491r1_SOL001ed421_resolution_of_CRD_related_editors_notes,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000495r2_SOL001Ed421_Minor_YAML_improvements,</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Information about changes</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>2021.10</td>
<td>4.0.8</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000097r9 Example_of_VNFD_in_support_of_OsContainers,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000440r2 SOL001ed421_editor_notes_handling_for_OsContainer,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000496r2 SOL001ed421_add_support_of_requirement_for_hugepages__,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000506 SOL001ed421_Removal_of_editor_s_note_on_bootdata,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000509 SOL001ed421_update_figure_6_1_2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000510 SOL001ed421_mapping_of_configurable_properties_in_Vdu_Compute</td>
</tr>
<tr>
<td>2021.10</td>
<td>4.0.9</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000439 OL001ed421_release_4_mirror_clarify_support_of_autoScale,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000448r4 SOL001ed421_Support_of_multiple_selectors_in_VnfPackageChange,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000525 SOL001ed421_Additional_YAML_fixes,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000528 SOL001ed421_Harmonisation_of_policy_target_specifications,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000529r1 SOL001ed421_Additional_Requirements_violation_in_Annex_A</td>
</tr>
<tr>
<td>2021.10</td>
<td>4.0.10</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000542r1 SOL001ed421_Rel__4_Mirror_of_475_and_541_Bugfix_with_adding_min_number_of_preserved_instances_to_NfviMaintenanceInfo</td>
</tr>
<tr>
<td>2021.11</td>
<td>4.0.11</td>
<td>Implemented:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000617r1 SOL001ed421_mciopProfile_note,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000618 SOL001ed421_Missing_references_to_note,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000619 SOL001ed421_Annex_A_18_corrections,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000621 SOL001ed421_note_reference_missing_for_affinityRule,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000622r1 SOL001ed421_monitoring_parameter_not_support_for_containers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000620r1 SOL001ed421_Sw_image_properties,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NFVSOL(21)000635 SOL001ed421_wrong_reference</td>
</tr>
</tbody>
</table>
## History

<table>
<thead>
<tr>
<th>Document history</th>
</tr>
</thead>
<tbody>
<tr>
<td>V4.2.1</td>
</tr>
</tbody>
</table>