The Standards People

VNF Descriptor (VNFD) Overview

Presented by:

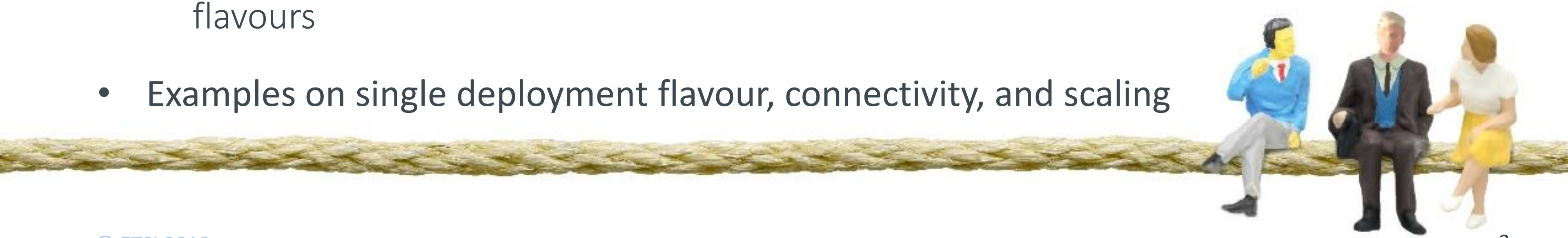
Thinh Nguyenphu, ETSI NFV SOL Vice-Chair, Nokia Bell Labs

Arturo Martin de Nicolas, Master Systems Designer, Ericsson

October 8, 2018

Topics

- VNF Descriptor (VNFD) Overview: NFV SOL 001 & status
  - Type definitions
  - OASIS TOSCA overview
  - Configurable properties & modifiable attributes
  - Type extension
  - VNFD service template design: single & multiple deployment flavours
- Examples on single deployment flavour, connectivity, and scaling



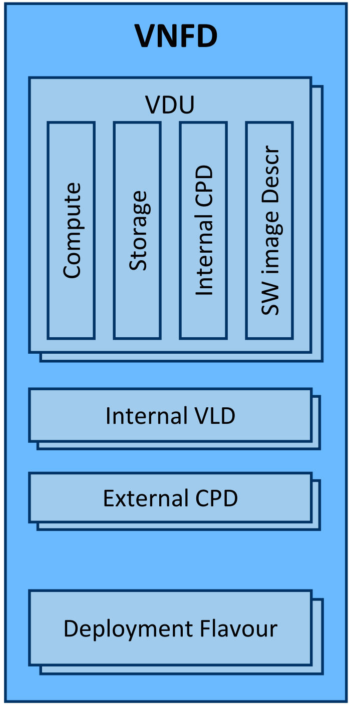
Part 1: VNFD Overview

VNF Descriptor (VNFD)

The VNFD defines VNF properties, such as:

- Resources needed (amount and type of Virtual Compute, Storage, Networking),
- Software metadata,
- Connectivity (descriptors for):
  - External Connection Points
  - Internal Virtual Links
  - Internal Connection Points
- Lifecycle management behavior (e.g. scaling, instantiation),
- Supported lifecycle management operations, and their configuration,
- Supported VNF specific parameters, and
- Affinity / anti-affinity rules.

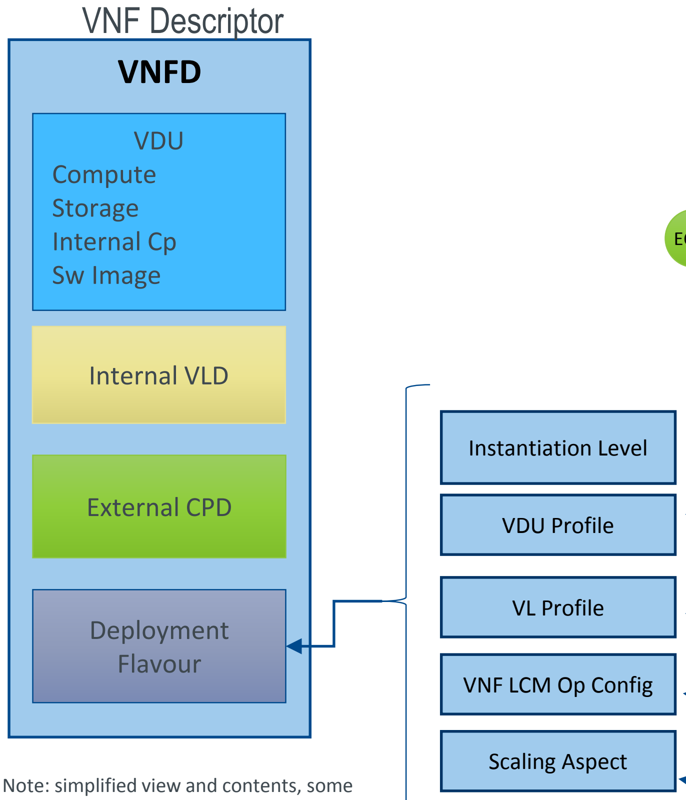
The VNFD defines deployment flavours (size-bounded deployment configurations, e.g. related to capacity).



Reference:
- ETSI GS NFV-IFA 011
- ETSI GS NFV-SOL 001*

* Pre-publication stage – drafts available

IFA011: VNF Descriptor (VNFD)



VNF Descriptor

VNFD

VDU
Compute
Storage
Internal Cp
Sw Image

Internal VLD

External CPD

Deployment Flavour

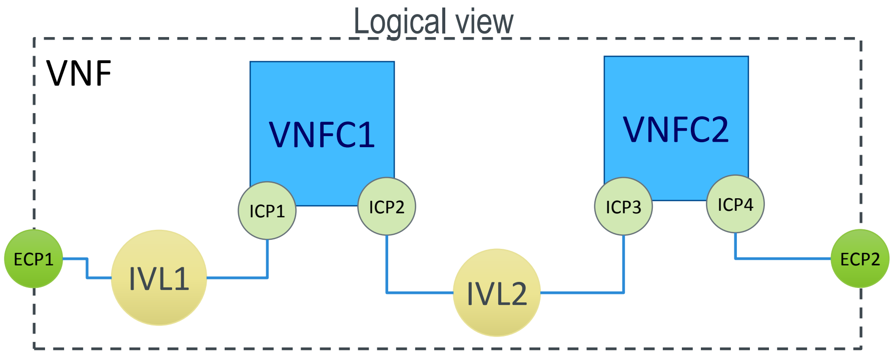
Instantiation Level

VDU Profile

VL Profile

VNF LCM Op Config

Scaling Aspect



Logical view

VNF

VNFC1

VNFC2

ICP1

ICP2

ICP3

ICP4

ECP1

ECP2

IVL1

IVL2

level of resources to be instantiated within a deployment flavour in term of the number VNFC instances to be created for each VDU

describes additional instantiation data for a given VDU.Compute used in the a specific deployment flavour.

additional instantiation data for a given VL used in a specific deployment flavour

represents information to configure lifecycle management operations

describes the details of an aspect used for horizontal scaling

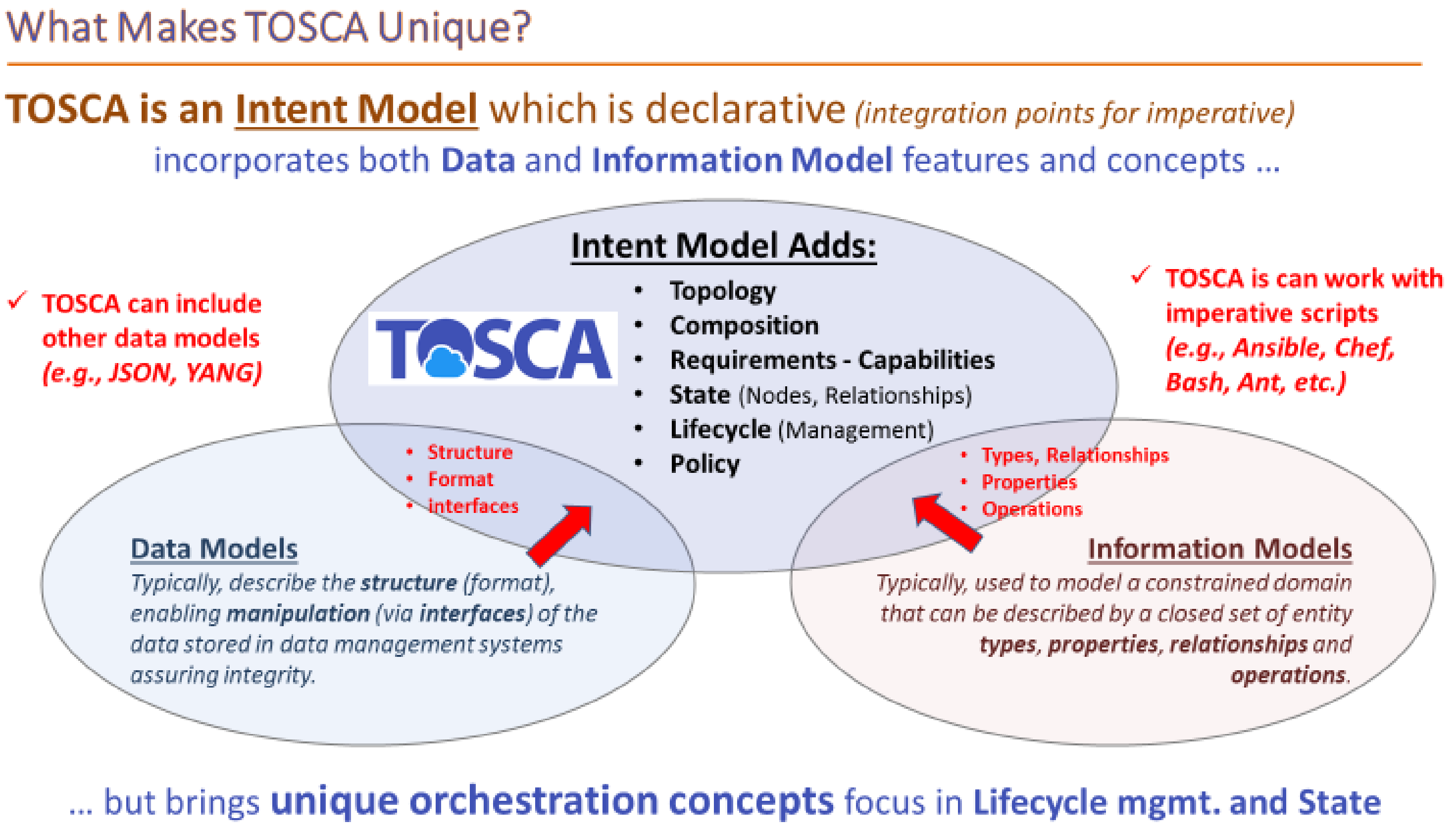
Note: simplified view and contents, some information elements are not illustrated

Part 2: OASIS TOSCA 101

ADD SECTION NAME

TOSCA 101:

Ref: OASIS TOSCA Overview



What Makes TOSCA Unique?

TOSCA is an Intent Model which is declarative *(integration points for imperative)*

incorporates both **Data** and **Information Model** features and concepts …

✓ TOSCA can include other data models *(e.g., JSON, YANG)*

Intent Model Adds:

- **Topology**
- **Composition**
- **Requirements - Capabilities**
- **State** (Nodes, Relationships)
- **Lifecycle** (Management)
- **Policy**

✓ TOSCA is can work with imperative scripts *(e.g., Ansible, Chef, Bash, Ant, etc.)*

- Structure
- Format
- interfaces

Data Models

*Typically, describe the **structure** (format), enabling **manipulation** (via **interfaces**) of the data stored in data management systems assuring integrity.*

- Types, Relationships
- Properties
- Operations

Information Models

*Typically, used to model a constrained domain that can be described by a closed set of entity **types**, **properties**, **relationships** and **operations**.*

… but brings **unique orchestration concepts** focus in **Lifecycle mgmt. and State**

TOSCA 101:

Ref: OASIS TOSCA Overview

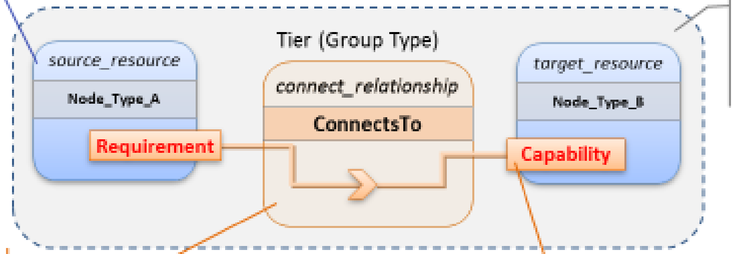
Topology – Define Topology with Nodes and Relationships

TOSCA is used first and foremost to describe the topology of the **deployment view** for cloud applications and services

- ✓ ***Node templates*** *to describe components in the topology structure*
- ✓ ***Relationship templates*** *to describe connections, dependencies, deployment ordering*

Nodes *- are the resources or components that will be materialized or consumed in the deployment topology*

Groups
Create Logical, Management or Policy groups (1 or more nodes)



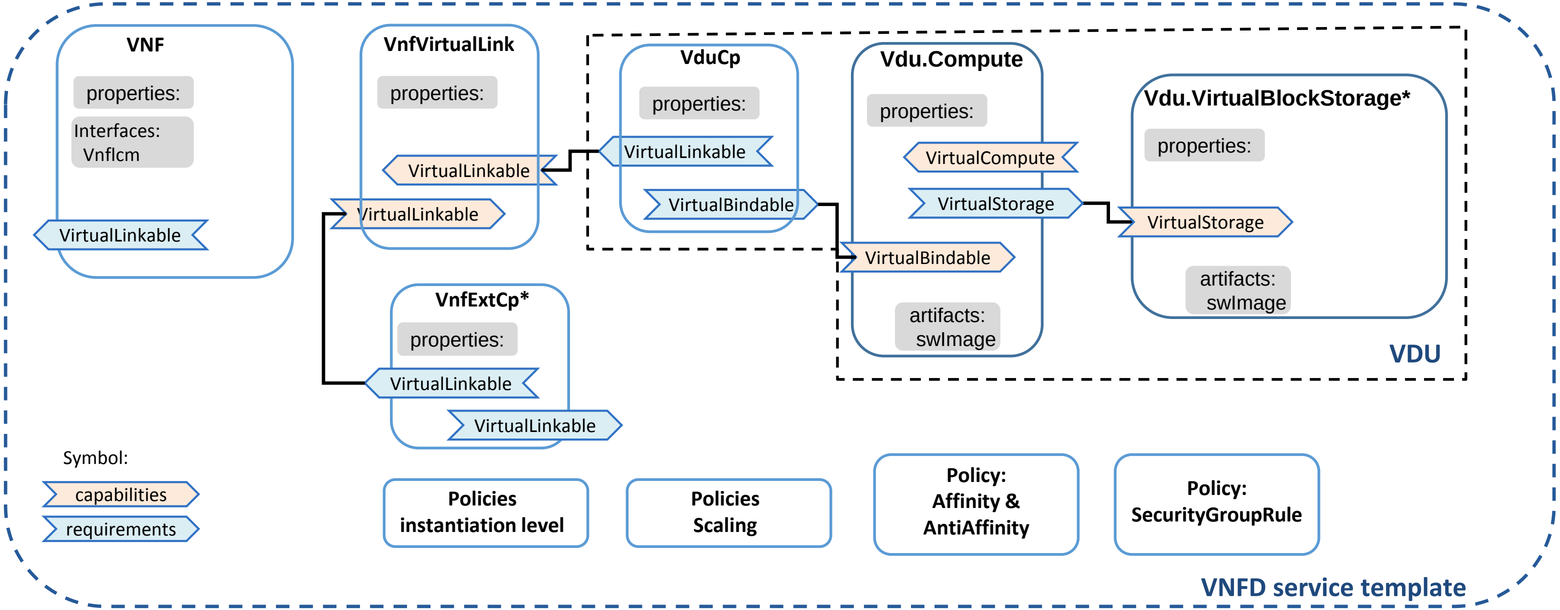
Relationships
express the dependencies between the nodes (not the traffic flow)

Requirement - Capability
Relationships can be customized to match specific source requirements to target capabilities

4

Part 3: NFV Type definitions Overview

SOL 001 (VNFD): TOSCA service template overview

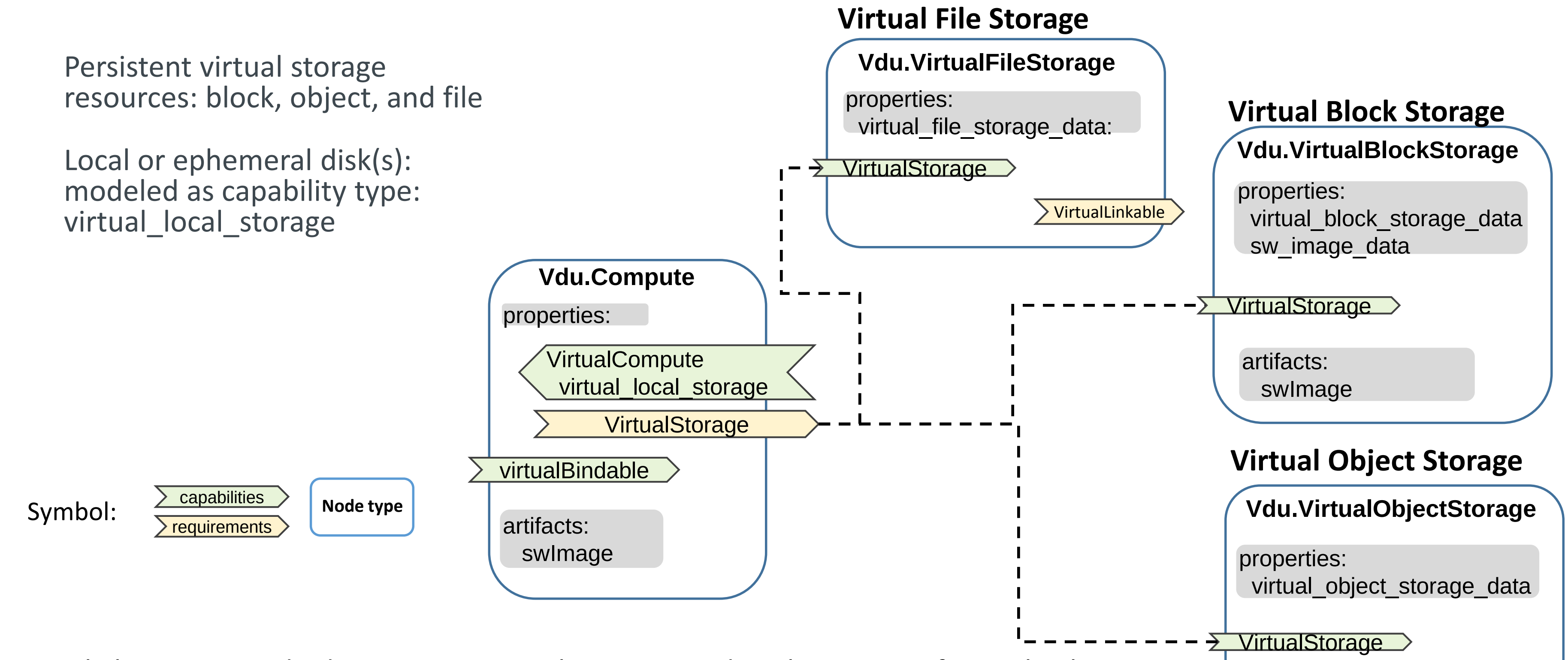


- SOL001 (v0.11.0): based on TOSCA Simple YAML Profile v1.2.
- https://docbox.etsi.org/ISG/NFV/Open/Drafts/SOL001_TOSCA_desc/NFV-SOL001v0110.zip
- In the case of single deployment flavour, SOL001 support both TOSCA Simple YAML Profile v1.1 and 1.2.
- Includes SOL001 NFV Type definition file: etsi_nfv_sol001_vnfd_2_5_1_types.yaml

Virtual Storage types:

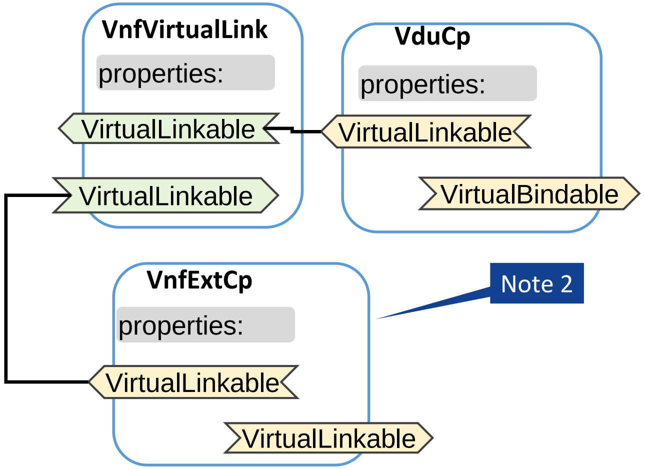
Persistent virtual storage resources: block, object, and file

Local or ephemeral disk(s): modeled as capability type: virtual_local_storage

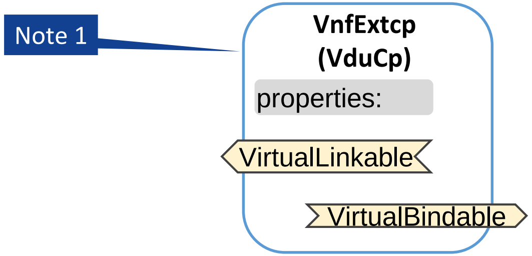


VirtualFileStorage and Vdu.Compute must be connected to the same vnfVirtualLink (not shown in the picture)

VNF external connection point



External CP connected to internal virtual link (VL)



External connection point (CP) re-exposing an internal CP

Note 1: In the case of re-exposing a VduCp as external connection point (VnfExtCp).

Note 2: 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.

- 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

VNF: configurable_properties & modifiable_attributes

configurable_properties: (tosca.datatypes.nfv.VnfConfigurableProperties): Describes the configurable properties of the VNF

- additionalConfigurableProperty: To be defined by the VNF provider to allow functional blocks such as the NFVO to instruct the VNFM to set certain properties of the VNF instance.
- must be declared in VNFD and default value may be declared. These default values will be part of the VnfInfo object, once the VNF instance is created.
- It must be set by Modify VNF Information operation via SOL003 or SOL002 API.
- Key point: this property is to be consume by VNF instance, via VNFM pass this information to VNF instance via Modify VNF Information operation.
- Applicable to tosca.datatypes.nfv.VnfcConfigurableProperties

modifiable_attributes: (tosca.datatypes.nfv.VnfInfoModifiableAttributes): Describes the modifiable attributes of the VNF

- These will allow an external functional block to set information that is typically for consumption by the VNFM, but is not always sent to the VNF instance.   It can be used by VNF LCM script.

Type extension

Type extension is used when VNF-specific type information is introduced in the VNFD: modifiable attributes, configurable properties, & additional parameters to LCM operations

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

Policy types

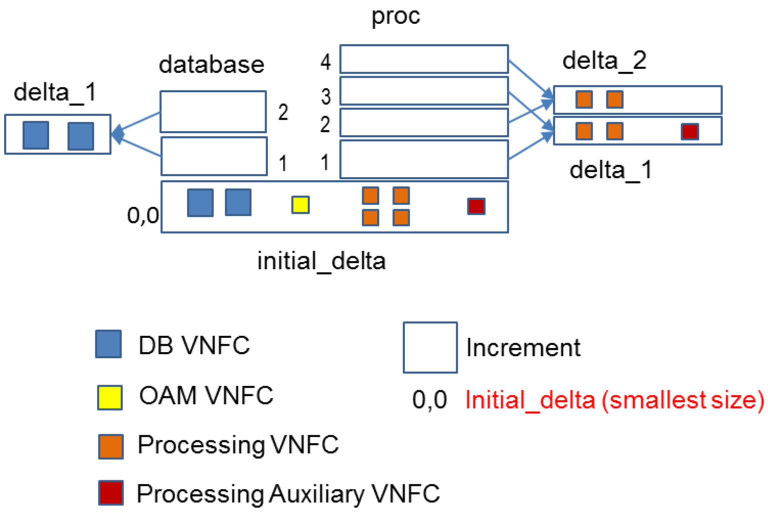
tosca.policies.nfv.InstantiationLevels
tosca.policies.nfv.VduInstantiationLevels
tosca.policies.nfv.VirtualLinkInstantiationLevels

tosca.policies.nfv.ScalingAspects
tosca.policies.nfv.VduScalingAspectDeltas
tosca.policies.nfv.VirtualLinkBitrateScalingAspectDeltas
tosca.policies.nfv.VduInitialDelta
tosca.policies.nfv.VirtualLinkBitrateInitialDelta

tosca.policies.nfv.AffinityRule

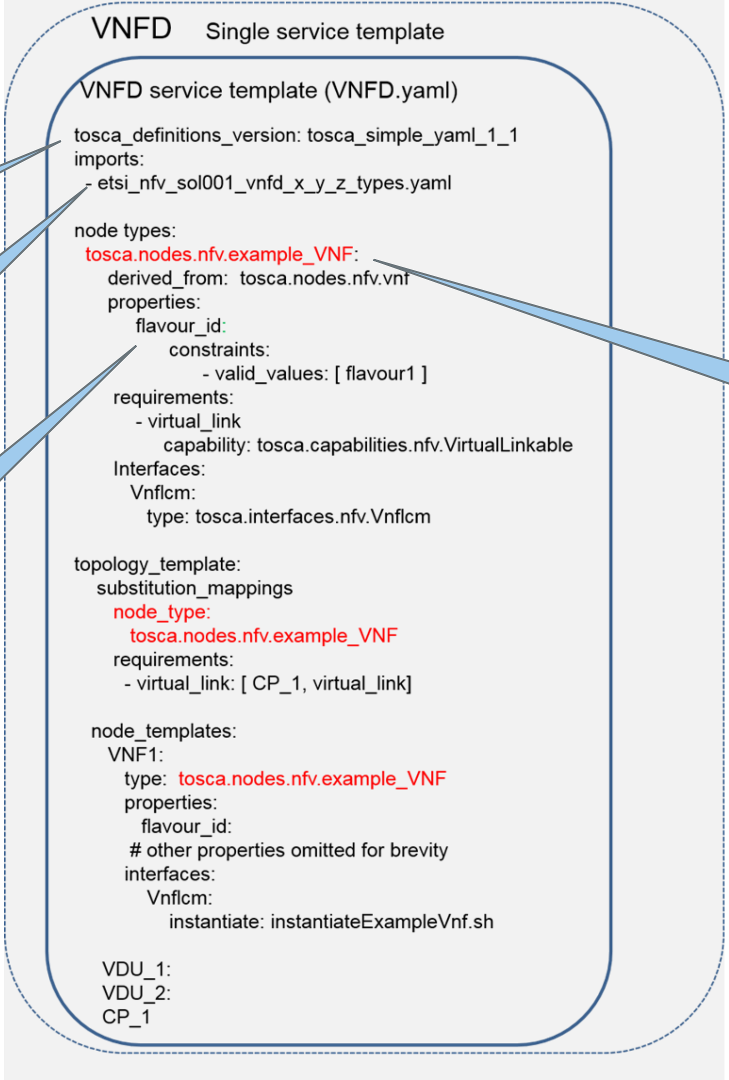
tosca.policies.nfv.SecurityGroupRule

Scaling model overview



Part 4: VNFD Service Template Design

1 service template design*: based on tosca_simple_yaml_1_1



VNFD Single service template

VNFD service template (VNFD.yaml)

```
tosca_definitions_version: tosca_simple_yaml_1_1
imports:
  - etsi_nfv_sol001_vnfd_x_y_z_types.yaml

node types:
 tosca.nodes.nfv.example_VNF:
    derived_from:  tosca.nodes.nfv.vnf
    properties:
        flavour_id:
            constraints:
                - valid_values: [ flavour1 ]
     requirements:
        - virtual_link
            capability: tosca.capabilities.nfv.VirtualLinkable
     Interfaces:
       Vnflcm:
          type: tosca.interfaces.nfv.Vnflcm

topology_template:
   substitution_mappings
     node_type:
       tosca.nodes.nfv.example_VNF
     requirements:
      - virtual_link: [ CP_1, virtual_link]

  node_templates:
    VNF1:
      type:  tosca.nodes.nfv.example_VNF
      properties:
        flavour_id:
       # other properties omitted for brevity
      interfaces:
         Vnflcm:
            instantiate: instantiateExampleVnf.sh

   VDU_1:
   VDU_2:
   CP_1
```

tosca_definitions_version : tosca_simple_yaml_1_1

Imports: SOL001 NFV types

flavor_id : required

VNF-specific node type: derived from tosca.nodes.nfv.vnf

*See backup slide for 2 Levels service template design

Part 5: SOL001 Status

Summary & next steps

VNFD part is stable

NSD part of specification is ongoing in ETSI NFV SOL WG (SOL001)

Work in progress:

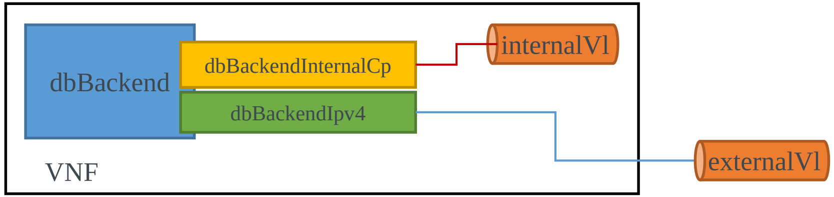
- Network Service Descriptor (NSD)

Work plan schedule	Stable Draft	Final Draft	WG App	TB App
SOL001 "TOSCA-based NFV descriptors spec"	2018.10.11	2018.11.08	2018.11.08	2018.12.08

Part 6: Examples

VNFD with single deployment flavour example

```
tosca_definitions_version: tosca_simple_yaml_v_1_1

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

topology_template:
 substitution_mappings:
  node_type: MyCompany.SunshineDB.1_0.1_0
  requirements:
   - virtual_link: [ dbBackendIpv4,   external_virtual_link ]

 node_templates:
  SunshineDB:
   type: MyCompany.SunshineDB.1_0.1_0
   properties:
    flavour_id: simple
    ...
   interfaces:
    Vnflcm:
     instantiate:
      implementation: instantiate.workbook.mistral.yaml
     ...
  mariaDbStorage:
   type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
    ...
   artifacts:
    sw_image:
```



- One deployment flavour
- Vdu.Compute node: dbBackend
- Two connection points: internal (dbBackendInternalCp); external (dbBackendIp4)
- One VNF internalVl and one externalVl

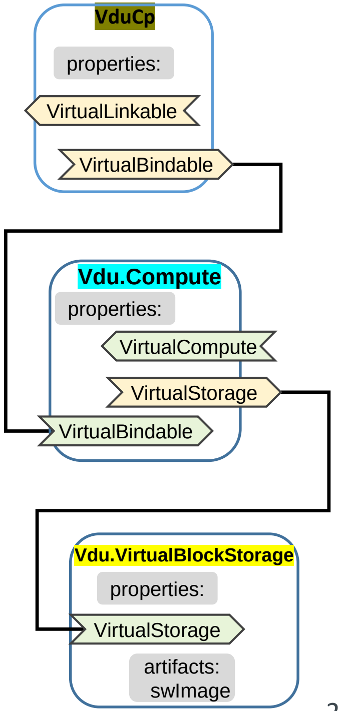
```
  dbBackend:
   type: tosca.nodes.nfv.Vdu.Compute
    ...
   capabilities:
    virtual_compute:
     ...
   requirements:
    - virtual_storage: mariaDb
Storage
  dbBackendInternalCp:
   type: tosca.nodes.nfv.VduCp
    ...
   requirements:
    - virtual_binding: dbBackend
    - virtual_link: internalVl
```

```
  internalVl:
   type: tosca.nodes.nfv.
VnfVirtualLink
    ...

  dbBackendIpv4:
   type: tosca.nodes.nfv.VduCp
    ...
   requirements:
    - virtual_link:
    - virtual_binding: dbBackend
```

SOL 001: Compute, storage, connection points

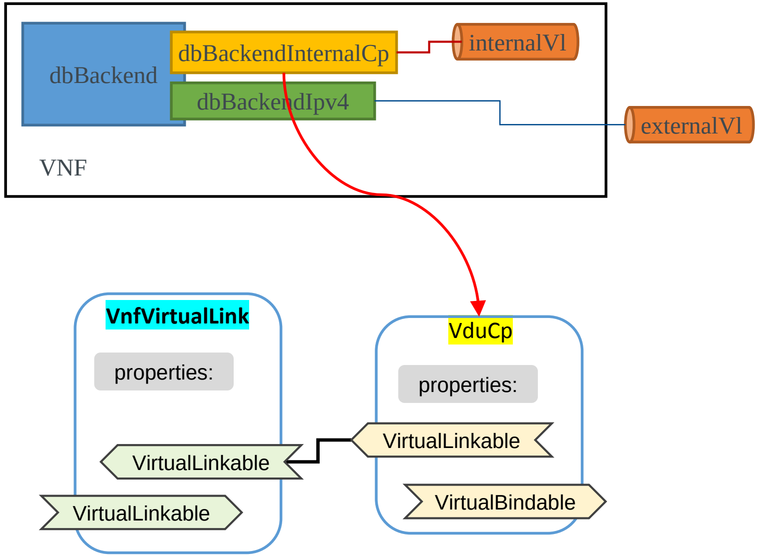
```
...

topology_template:

...
  node_templates:
    dbBackend:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: ..
        description: ..
        boot_order: ..
        nfvi_constraints: ..
        configurable_properties:
          additional_vnfc_configurable_properties: {}
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 4
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 8096 MB
            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: ..
           rdma_enabled:  ..
         sw_image_data:
           name: Software of Maria Db
           version: 1.0
           checksum: 9af30fce37a4c5c831e095745744d6d2
           container_format: bare
           disk_format: qcow2
           min_disk: 2 GB
           min_ram: 8096 MB
           size: 2 GB
           operating_system: Linux
           supported_virtualisation_environments:
             - KVM
       artifacts:
         sw_image:
           type: tosca.artifacts.nfv.SwImage
           file: maria.db.image.v1.0.qcow2
    dbBackendInternalCp:
      type: tosca.nodes.nfv.VduCp
       properties:
         layer_protocols: [ ipv4 ]
         role: leaf
         description: Internal connection point on an VL
         protocol_data: [ associated_layer_protocol: ipv4 ]
         trunk_mode: false
      requirements:
         - virtual_binding: dbBackend
         - virtual_link: internalVl
```



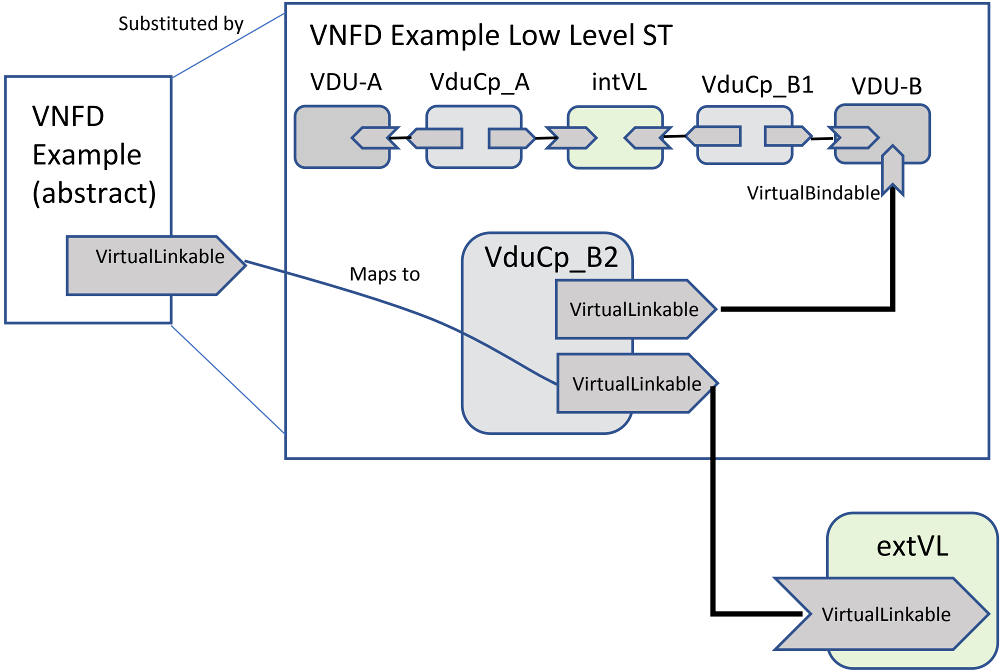
Internal connectivity

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

    dbBackendInternalCp:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv4 ]
        role: leaf
        description: Internal connection point on an VL
        protocol_data: [ associated_layer_protocol: ipv4 ]
        trunk_mode: false
      requirements:
        - virtual_binding: dbBackend
        - virtual_link: internalVl

    internalVl:
      type: tosca.nodes.nfv.VnfVirtualLink
      properties:
        connectivity_type:
          layer_protocols: [ ipv4 ]
          flow_pattern: mesh
        test_access: []
        description: ..
        vl_profile:
          qos:
          maxBitRateRequirements:
          minBitRateRequirements:
      capabilities:
        virtual_linkable:
```



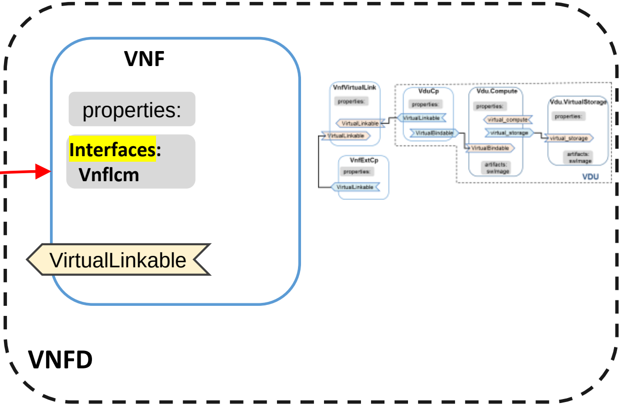
External connectivity: re-exposing a VduCp

```
tosca_definitions_version: tosca_simple_yaml_v_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ...
    requirements:
      - virtual_link:
          capability:
             tosca.capabilities.nfv.VirtualLinkable

topology_template:

   substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    requirements:
      - virtual_link: [vduCp_B2, virtual_link]

  node_templates:

    vduCp_B2:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv4 ]
        role: leaf
        description: External connection point
      requirements:
        - virtual_binding: VDU-B
        - virtual_link:
```



VNF Lifecycle management (LCM) interfaces

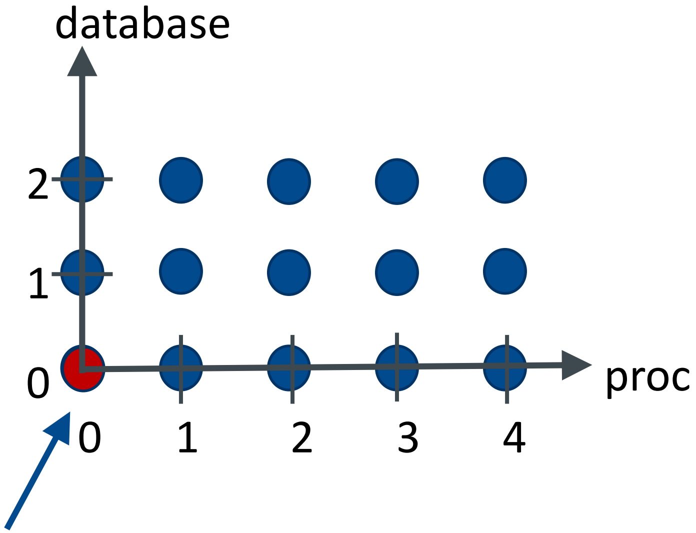
```
...
data_types:

  MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters:
    derived_from: tosca.datatypes.nfv.VnfOperationAdditionalParameters
    properties:
      parameter_1:
        type: string
        required: false
        default: value_1
      parameter_2:
        type: string
        required: false
        default: value_2
node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
    interfaces:
      Vnflcm:
        instantiate:
          inputs:
            instantiate_additional_parameter:
              type:
MyCompany.datatypes.nfv.VnfInstantiateAdditionalParameters
              required: true
        terminate:

topology_template:

  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    requirements:
      - virtual_link: [ dbBackendIpv4,   external_virtual_link ]
```



```
node_templates:
  SunshineDB:
    type: MyCompany.SunshineDB.1_0.1_0
    interfaces:
      Vnflcm:
        instantiate:
          implementation: instantiate.workbook.mistral.yaml
        terminate:
          implementation: terminate.workbook.mistral.yaml

  dbBackend:
    type: tosca.nodes.nfv.Vdu.Compute

  mariaDbStorage:
    type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
 ...
```

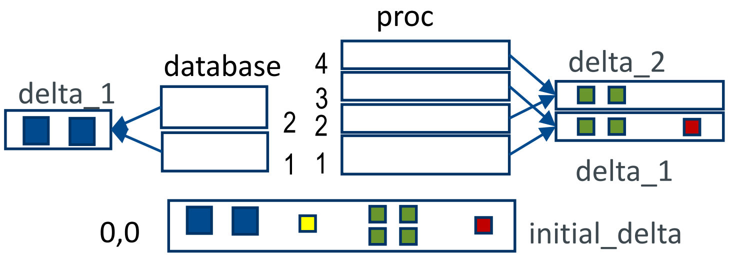
InstantiationLevels and ScalingAspects policies

External view: Scaling aspects



Zero point (“smallest size”) = scale level 0

Example VNF internal view: Groups of VNFCs



🟦 DB VNFC

🟨 OAM VNFC

🟩 Processing VNFC

🟥 Processing Auxiliary VNFC

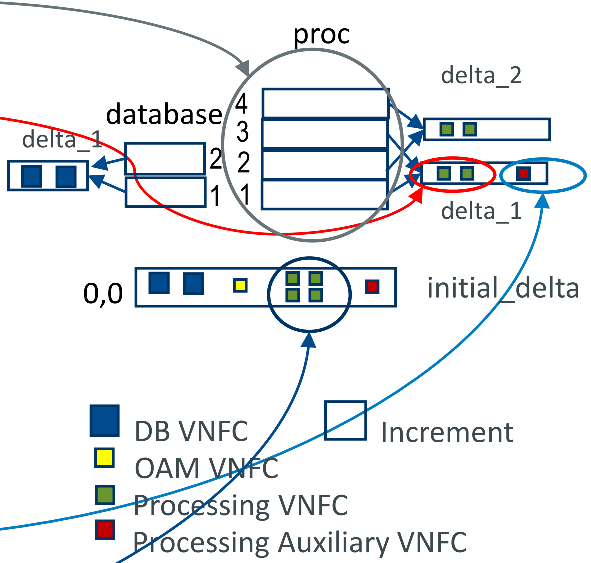
□ Increment

0,0 Initial increment (smallest size)

InstantiationLevels and ScalingAspects 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
- 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 ]
```

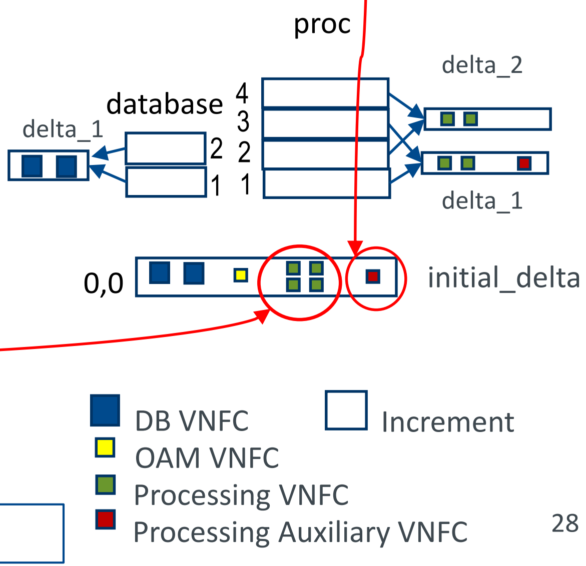


InstantiationLevels and ScalingAspects policies

```
 ...
node_templates:

  db:
   type: tosca.nodes.nfv.VDU.Compute
    properties:
      vdu_profile:
        min_number_of_instances: 2
        max_number_of_instances: 6
    ...
  oam:
   type: tosca.nodes.nfv.VDU.Compute
    properties:
      vdu_profile:
        min_number_of_instances: 1
        max_number_of_instances: 1
    ...
  processing:
   type: tosca.nodes.nfv.VDU.Compute
    properties:
      vdu_profile:
        min_number_of_instances: 4
        max_number_of_instances: 12
    ...
  processing_auxiliary:
   type: tosca.nodes.nfv.VDU.Compute
    properties:
      vdu_profile:
        min_number_of_instances: 1
        max_number_of_instances: 3
    ...
```

```
policies:
 - instantiation_levels:
     type: tosca.policies.nfv.InstantiationLevels
     properties:
       levels:
         instantiation_level_1:
           description: ..
           scale_info:
             database:
               scale_level: 0
             proc:
               scale_level: 0
         instantiation_level_2:
           description: ..
           scale_info:
             database:
               scale_level: 1
             proc:
               scale_level: 1
          default_level: instantiation_level_1

- processing_instantiation_levels:
      type:
tosca.policies.nfv.VduInstantiationLevels
      properties:
        levels:
          instantiation_level_1:
            number_of_instances: 4
          instantiation_level_2:
            number_of_instances: 6
        targets: [ processing ]
```

```
 - processing_auxiliary_instantiation_levels:
     type: tosca.policies.nfv.VduInstantiationLevels
     properties:
       levels:
         instantiation_level_1:
           number_of_instances: 1
         instantiation_level_2:
           number_of_instances: 2
       targets: [ processing_auxiliary ]

 ...
```



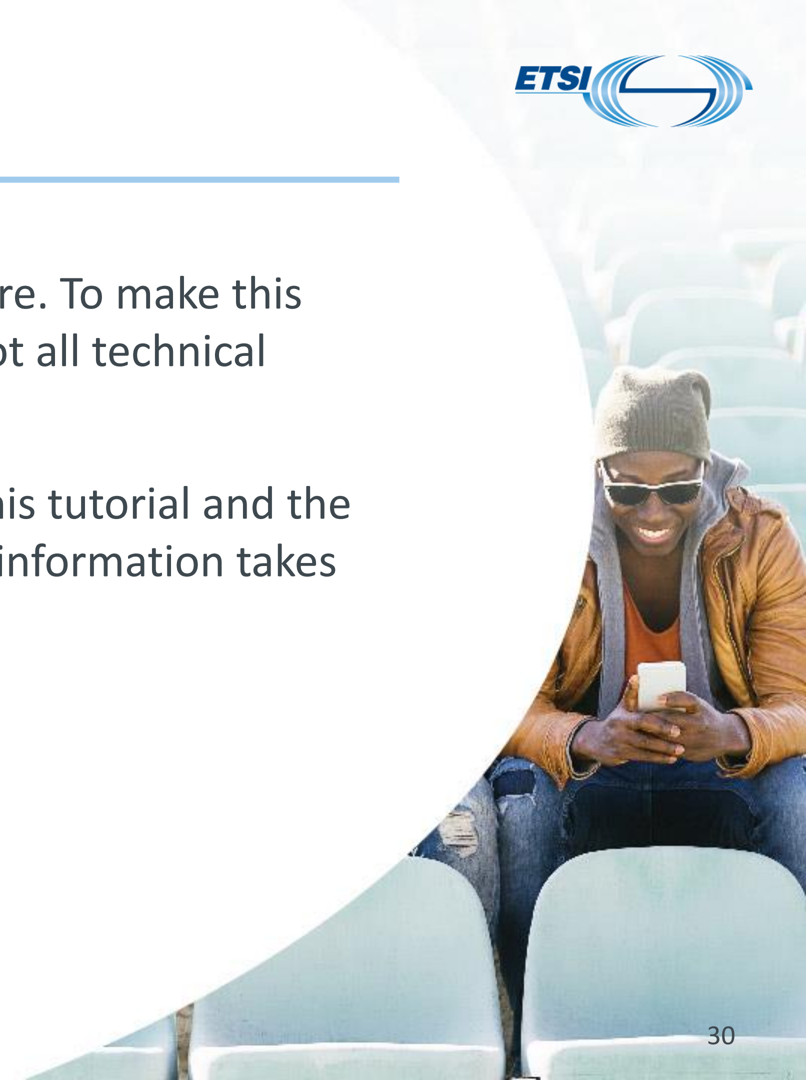
Note: example in SOL001 has only one instantiation level

Thank You!

DISCLAIMER

The contents of this presentation is of tutorial nature. To make this presentation easy to understand to non-experts, not all technical details are shown.

In case of discrepancies between the contents of this tutorial and the ETSI NFV Group Specifications, the latter source of information takes precedence.

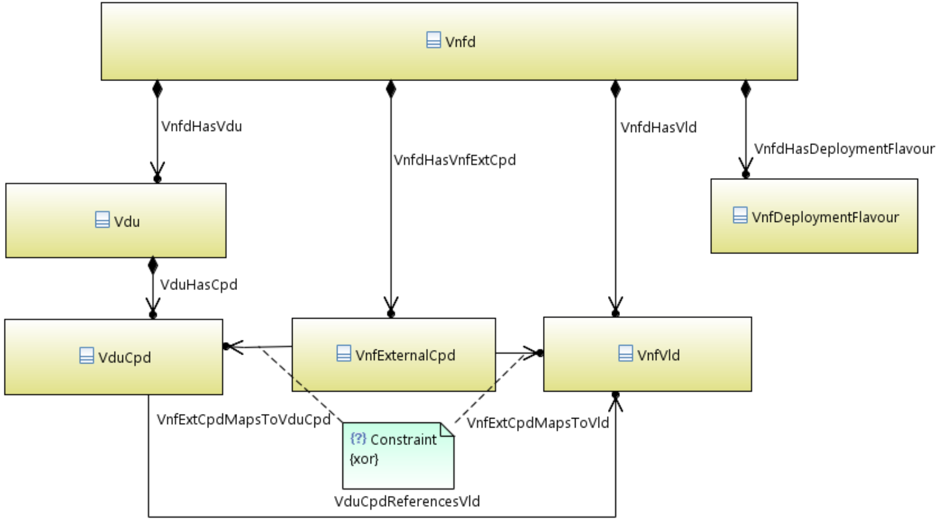


Backup

Mapping IFA 011 elements with TOSCA types

Mapping of IFA 011 information elements with TOSCA types

IFA 011 Elements	VNFD TOSCA types	Derived from
VNFD	tosca.nodes.nfv.VNF	tosca.nodes.Root
Vdu	n/a	n/a
Cpd (Connection Point)	tosca.nodes.nfv.Cp	tosca.nodes.Root
VduCpd (internal connection point)	tosca.nodes.nfv.VduCp	tosca.nodes.nfv.Cp
VnfVirtualLinkDesc (Virtual Link)	tosca.nodes.nfv.VnfVirtualLink	tosca.nodes.Root
VnfExtCpd (External Connection Point)	tosca.nodes.nfv.VnfExtCp tosca.nodes.nfv.VduCp	tosca.nodes.nfv.Cp
Virtual Storage	tosca.nodes.nfv.Vdu.VirtualBlockStorage tosca.nodes.nfv.Vdu.VirtualObjectStorage tosca.nodes.nfv.Vdu.VirtualFileStorage	tosca.nodes.Root
Virtual Compute	tosca.nodes.nfv.Vdu.Compute	tosca.nodes.Root
Software Image	tosca.artifacts.nfv.SwImage	tosca.artifacts.Deployment.Image
Deployment Flavour	Represented as a TOSCA service template	n/a
Scaling	Policy types	tosca.nodes.Root
Instantiation Level	Policy types	tosca.nodes.Root



IFA 011 information model

- ➢ SOL001 (v0.11.0): based on TOSCA Simple YAML Profile v1.2.
- ➢ In the case of single deployment flavour, SOL001 support both TOSCA Simple YAML Profile v1.1 and 1.2.

- ➢ SOL001 (v0.11.0) is a **stable draft (VNFD part)**,
- ➢ https://docbox.etsi.org/ISG/NFV/Open/Drafts/SOL001_TOSCA_desc/NFV-SOL001v0110.zip

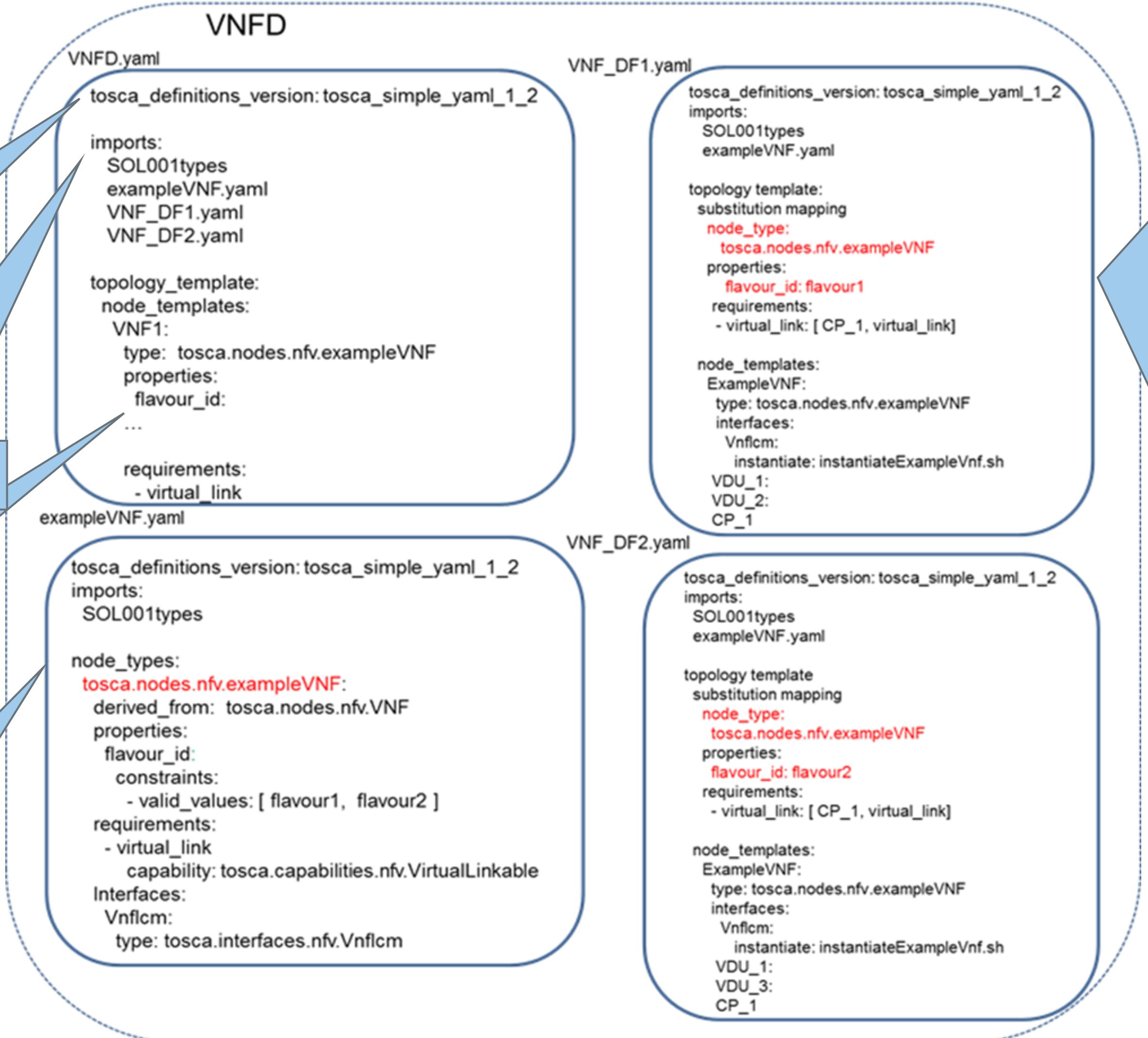
2 Levels service template design: key points

tosca_definitions_version : tosca_simple_yaml_1_2

Imports: lower STs, VNF specific, and SOL001 types

flavor_id : required

Separate ST for VNF specific



VNFD.yaml

```
tosca_definitions_version: tosca_simple_yaml_1_2

imports:
  SOL001types
  exampleVNF.yaml
  VNF_DF1.yaml
  VNF_DF2.yaml

topology_template:
  node_templates:
    VNF1:
      type:  tosca.nodes.nfv.exampleVNF
      properties:
        flavour_id:
      …

      requirements:
       - virtual_link
```

exampleVNF.yaml

```
tosca_definitions_version: tosca_simple_yaml_1_2
imports:
  SOL001types

node_types:
  tosca.nodes.nfv.exampleVNF:
    derived_from:  tosca.nodes.nfv.VNF
    properties:
      flavour_id:
        constraints:
          - valid_values: [ flavour1,  flavour2 ]
    requirements:
      - virtual_link
          capability: tosca.capabilities.nfv.VirtualLinkable
    Interfaces:
      Vnflcm:
        type: tosca.interfaces.nfv.Vnflcm
```

VNF_DF1.yaml

```
tosca_definitions_version: tosca_simple_yaml_1_2
imports:
  SOL001types
  exampleVNF.yaml

topology template:
  substitution mapping
    node_type:
      tosca.nodes.nfv.exampleVNF
    properties:
      flavour_id: flavour1
    requirements:
     - virtual_link: [ CP_1, virtual_link]

  node_templates:
    ExampleVNF:
      type: tosca.nodes.nfv.exampleVNF
      interfaces:
        Vnflcm:
          instantiate: instantiateExampleVnf.sh
    VDU_1:
    VDU_2:
    CP_1
```

VNF_DF2.yaml

```
tosca_definitions_version: tosca_simple_yaml_1_2
imports:
  SOL001types
  exampleVNF.yaml

topology template
  substitution mapping
    node_type:
      tosca.nodes.nfv.exampleVNF
    properties:
      flavour_id: flavour2
    requirements:
     - virtual_link: [ CP_1, virtual_link]

  node_templates:
    ExampleVNF:
      type: tosca.nodes.nfv.exampleVNF
      interfaces:
        Vnflcm:
          instantiate: instantiateExampleVnf.sh
    VDU_1:
    VDU_3:
    CP_1
```

Lower level ST: implementable TOSCA ST for deployment specific DF:

topology template describing the internal topology of the VNF with: substitution_mappings indicating:

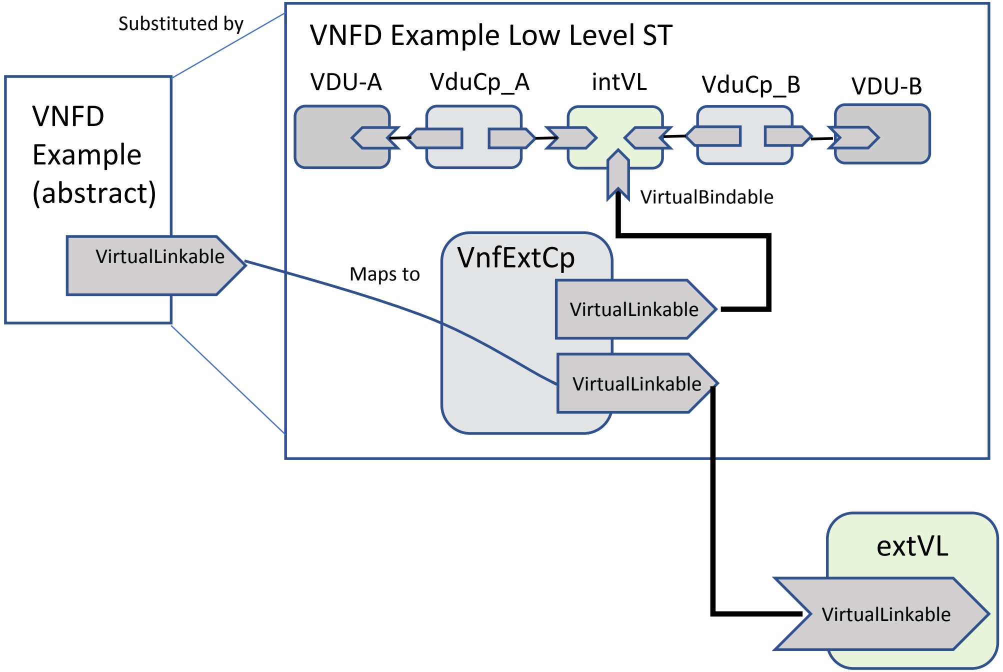
a) the same node type as defined in the top level service template,

b) a flavour_id property value which identifies this DF within the VNFD
c) the mapping of the virtual_link requirement

Implementations of the LCM interfaces of the VNF

External connectivity: external connection point connected to an internal virtual link

```
...
node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      ...
    requirements:
      - virtual_link:
          capability:
             tosca.capabilities.nfv.VirtualLinkable

topology_template:

  substitution_mappings:
    node_type: tosca.nodes.nfv.exampleVNF
    requirements:
      - virtual_link: [myMRFExtCp, external_virtual_link]

  node_templates:

    mrfExtCp :
      type: tosca.nodes.nfv.VnfExtCp
      # properties omitted
      requirements:
        - external_virtual_link:
        - internal_virtual_link:  intVL-A

   intVL-A:
     type: tosca.nodes.nfv.VnfVirtualLink
     properties:
       ...
   capabilities:
     virtual_linkable
```



Substituted by
VNFD Example Low Level ST
VDU-A
VduCp_A
intVL
VduCp_B
VDU-B
VNFD Example (abstract)
VirtualBindable
VirtualLinkable
Maps to
VnfExtCp
VirtualLinkable
VirtualLinkable
extVL
VirtualLinkable