VNF DESCRIPTOR (VNFD) & VNF PACKAGE OVERVIEW

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

April 24, 2018

NFV & ZERO TOUCH WORLD CONGRESS

Topics

- VNF Descriptor (VNFD) Overview: NFV SOL 001
- VNF Package Overview: NFV SOL 004

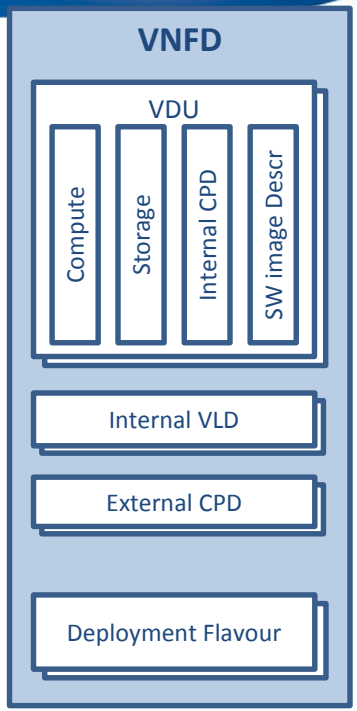
PART 1

VNF Descriptor (VNFD) Specification: NFV SOL 001

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.

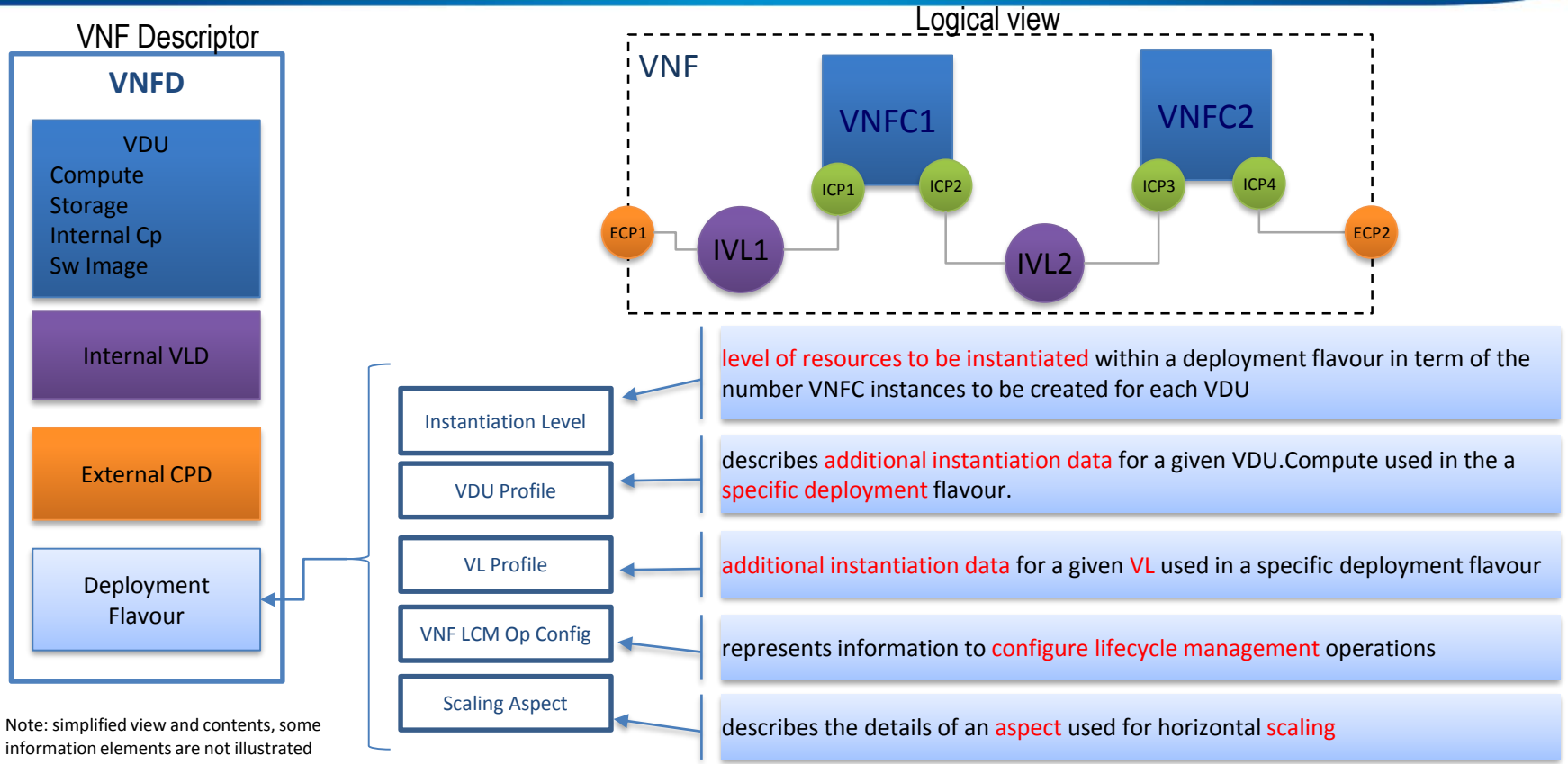
- The VNFD defines deployment flavours (size-bounded deployment configurations, e.g. related to capacity).



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

* Pre-publication stage – drafts available

VNF Descriptor (VNFD)

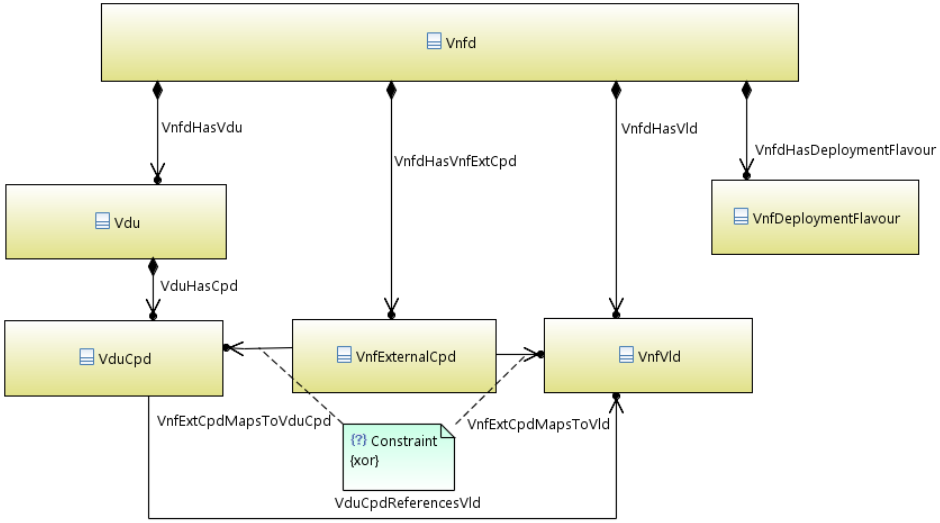


Note: simplified view and contents, some information elements are not illustrated

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.Cp
Virtual Storage	tosca.nodes.nfv.Vdu.VirtualStorage	tosca.nodes.Root
Virtual Compute	tosca.nodes.nfv.Vdu.Compute	tosca.capabilities.Root
Software Image	tosca.artifacts.nfv.SwImage	tosca.artifacts.Deployment.Image
Deployment Flavour	TBD	TBD
Scaling Aspect	TBD	TBD
Element Group	tosca.groups.nfv.ElementGroup	tosca.nodes.Root
Instantiation Level	TBD	TBD

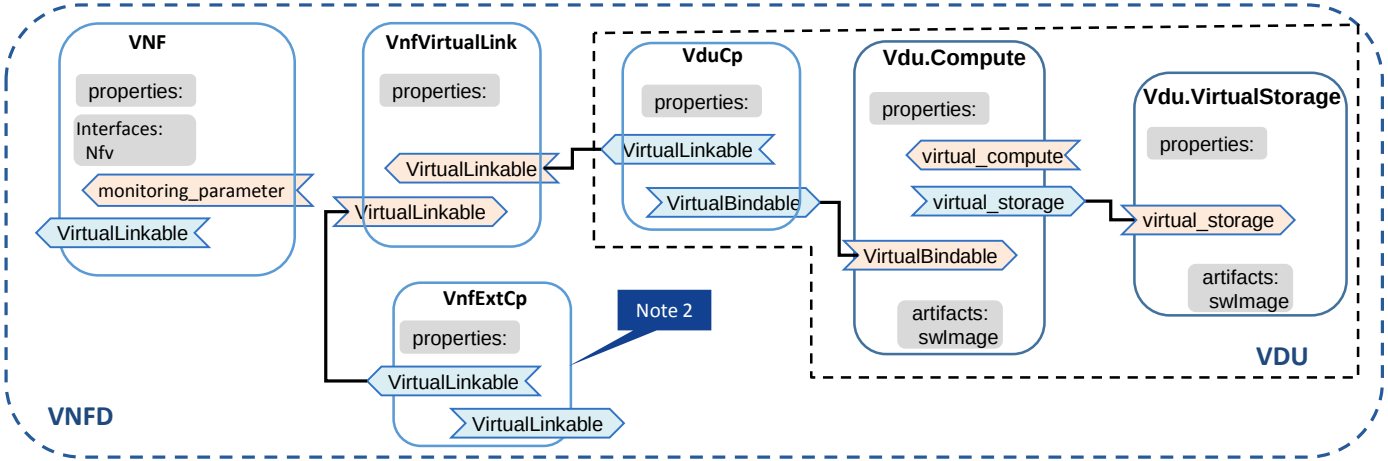


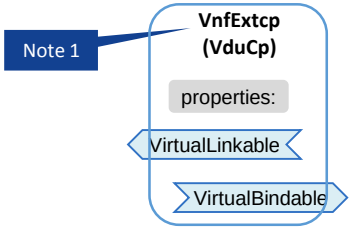
IFA 011 information model

- SOL001 (v0.6.1) is **working draft**, expecting a stable draft (VNFD part) by mid May 2018.
- https://docbox.etsi.org/ISG/NFV/Open/Drafts/SOL001_TOSCA_desc/NFV-SOL001v061.zip
- The content of this presentation is based on SOL011 v0.6.1

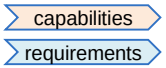
- SOL001 (v0.6.1): based on TOSCA Simple YAML Profile v1.2.
- In the case of single deployment flavour, SOL001 support both TOSCA Simple YAML Profile vv1.1 and 1.2.

SOL 001: TOSCA service template overview





Symbol:



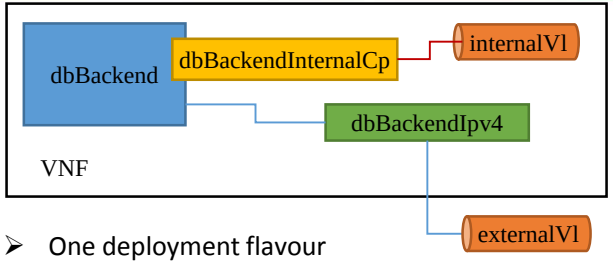
Note 1: In the case of re-exposing a VduCp as external connection point (VnfExtCp).
Note 2: A node template of this type is used to represent a VNF external connection point only in the case the VnfExtCp is connected to an internal virtual link.

- 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

Namespace Prefix	Specification Description
toscanfv	The reserved TOSCA prefix that can be associated with the TOSCA Namespace URI as declared in the present document.

VNFD: Example

```
tosca_definitions_version: tosca_simple_profile_yaml v 1.1

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      descriptor_id:
      provider:
      product_name:
      software_version:
      descriptor_version:
      flavour_id:
      ...
    interfaces:
      Nfv:

topology_template:
  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    requirements:
      - virtual_link: [ dbBackendIpv4,   external_virtual_link ] # IPv4 for SQL

  node_templates:
    dbBackend:
      type: tosca.nodes.nfv.Vdu.Compute
        ...
      capabilities:
        virtual_compute:
          ...
      requirements:
        - virtual_storage: mariaDbStorage

    mariaDbStorage:
      type: tosca.nodes.nfv.Vdu.VirtualStorage
        ...
      artifacts:
        sw_image:
```



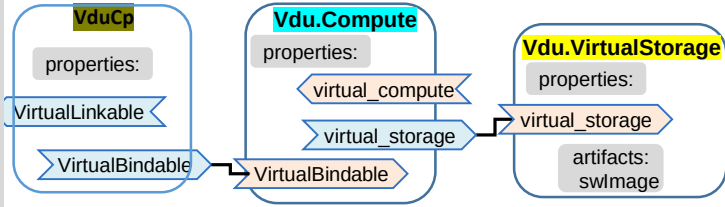
- One deployment flavour
- Vdu.Compute node: dbBackend
- Two connection points: internal (dbBackendInternalCp); externa (dbBackendIp4)
- Two virtual links: internalVl and externalVl

```
dbBackendInternalCp:
  type: tosca.nodes.nfv.Cp
    ...
  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

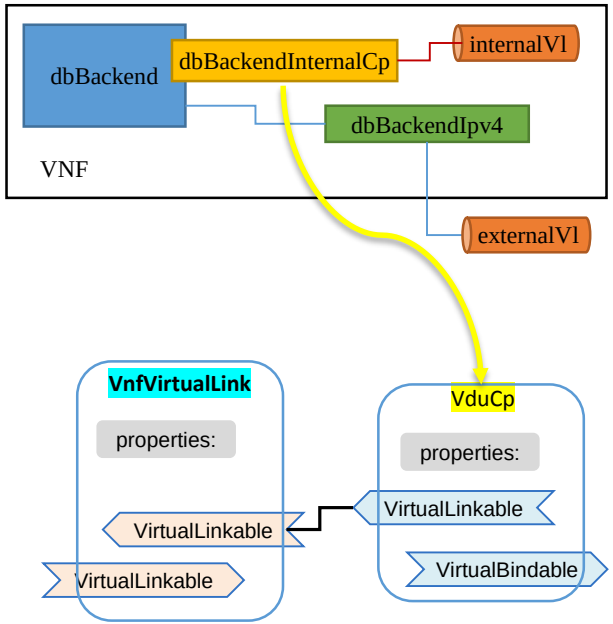
```
tosca_definitions_version:
tosca_simple_profile_yaml v 1.1

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

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.VirtualStorage
    properties:
      type_of_storage: ..
      size_of_storage: ..
      rdma_enabled:  ..
      sw_image_data:
        name: Software of Maria Db
        version: 1.0
        checksum: 9af30fce37a4c5c831e095745744d6d2
        container_format: qcow2
        disk_format: bare
        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.Cp
      properties:
        layer_protocol: ipv4
        role: leaf
        description: Internal connection point on an VL
      requirements:
       - virtual_binding: dbBackend
       - virtual_link: internalVl
```



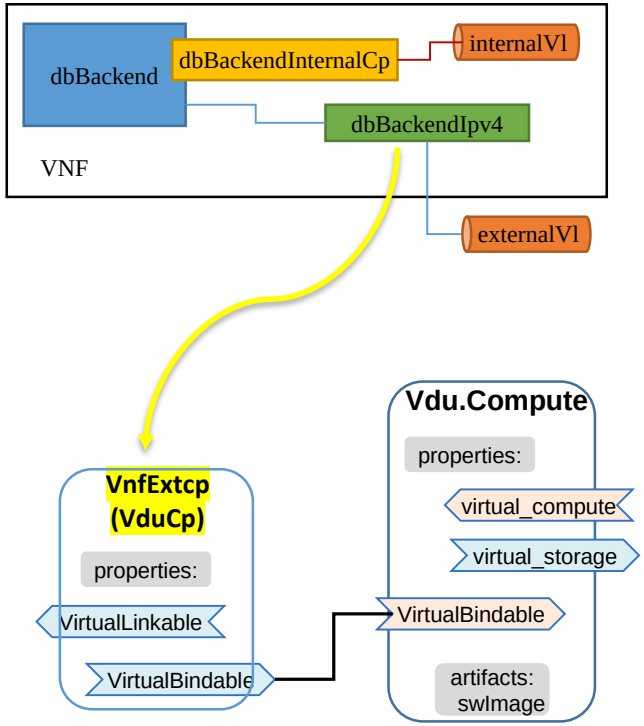
Internal connectivity

```
tosca_definitions_version: tosca_simple_profile_yaml v 1.1

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

topology_template:


  node_templates:

    dbBackendInternalCp:
      type: tosca.nodes.nfv.Cp
      properties:
        layer_protocol: 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_protocol: ipv4
          flow_pattern: mesh
        test_access: []
        description: ..
        vl_profile:
          qos:
          maxBitRateRequirements:
          minBitRateRequirements:
```



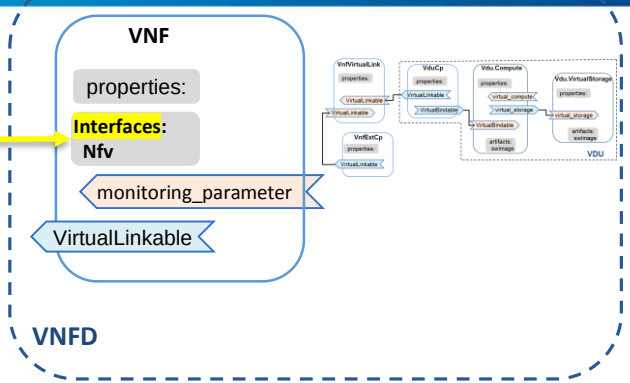
External connectivity (Case: re-exposing a VduCp)

```
tosca_definitions_version: tosca_simple_profile_yaml v 1.1

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

topology_template:


  node_templates:

    dbBackendIpv4:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocol: ipv4
        role: leaf
        description: External connection point to access the DB on IPv4
      requirements:
        - virtual_link:
        - virtual_binding: dbBackend
```



VNF LCM “event” and “scripts”

```
tosca_definitions_version: tosca_simple_profile_yaml v 1.1

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
    interfaces:
      Nfv:
        instantiate:
          inputs:
            parameter_1:
              type: string
              required: false
              default: value_1
            parameter_2:
              type: string
              required: false
              default: value_2
        terminate:
          implementation: terminate.workbook.mistral.yaml

topology_template:

  substitution_mappings:
    node_type: MyCompany.SunshineDB.1_0.1_0
    requirements:
      - virtual_link: [ dbBackendIpv4,   external_virtual_link ] #
IPv4 for SQL
```




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

  mariaDbStorage:
    type: tosca.nodes.nfv.Vdu.VirtualStorage

  dbBackendInternalCp:
    type: tosca.nodes.nfv.Cp

  internalVl:
    type: tosca.nodes.nfv.VnfVirtualLink

  dbBackendIpv4:
    type: tosca.nodes.nfv.VduCp
```

Summary & next steps

- VNFD specification ongoing in ETSI NFV SOL WG (SOL001)
- Work in progress:
  - DeploymentFlavour: InstantiationLevel, ScaleInfo
  - ElementGroup, Affinity/AntiAffinity, LocalAffinity/AntiAffinityRule
  - VnfIndicator and monitoring
  - AutoScale
  - Network Service Descriptor (NSD)

Work plan schedule	Stable Draft	Final Draft	WG App	TB App
SOL001 "TOSCA-based NFV descriptors spec"	2018.07.01	2018.08.26	2018.08.31	2018.09.30

  - Stable Draft milestone for the VNFD part of SOL001 = **2018.05.17**
- Way forward: join forces in standardization (ETSI NFV & OASIS TOSCA).

PART 2

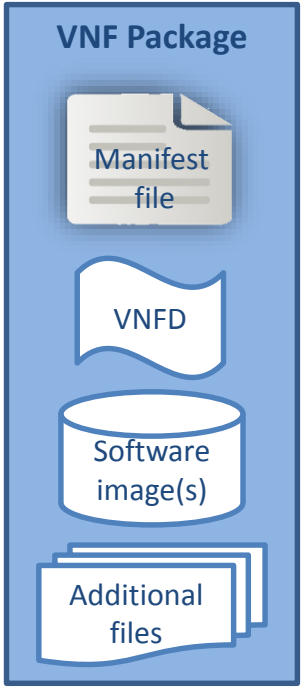
VNF PACKAGE SPECIFICATION (NFV SOL 004)

Why VNF Package Standard Is Needed?

- There is a need for a uniform way for VNF providers to deliver VNFs to service providers and making the delivery simple, effective and efficient
- TOSCA YAML CSAR (Cloud Service Archive) is a good basis for VNF packaging but it lacks important telco grade functionality such as security
- A VNF package standard
  - Enables automatic execution of VNF on-boarding and acceptance testing
  - Mandates VNFs to be interoperable with independently developed NFV management and orchestration systems
  - Add to CSAR means for validating package integrity and authenticity

Packaging a VNF: VNF Package

- The VNF Package contains:
  - the VNF descriptor (VNFD) that defines metadata for package onboarding and VNF management,
  - the software images needed to run the VNF, and
  - Manifest file that provides package integrity and authenticity
  - (optional) additional files to manage the VNF (e.g. scripts, vendor-specific files etc.).
- The VNF Package is delivered by the VNF provider as a whole and is immutable (protected from modification).
- The VNF Package or its Manifest file is digitally signed
- The VNF Package is stored in a repository by the NFVO.
- The VNF Package can be accessed by VNFM.



Reference:
- ETSI GS NFV-IFA 011
- ETSI GS NFV-SOL 004

VNF Package format

- A VNF Package is a Cloud Service ARchive (CSAR)
- A CSAR file is a ZIP file with a well-defined structure.
  - The structure and format of a VNF package shall conform to the TOSCA Simple Profile YAML v1.1/v1.2 Specification of the CSAR format.
- The VNFD is the main TOSCA definitions YAML file inside the archive.

References:
- TOSCA-Simple-Profile-YAML-v1.1 and v1.2

VNF Package
Standard Artifacts and Directories

- Change History File
  - Humanly readable text file
  - All the changes in the VNF package shall be versioned, tracked and inventoried
- Testing Files
  - Goal is to enable VNF package validation
  - VNF Provider includes files containing necessary information (e.g. test description)
- Licensing Information for released VNF
  - Include a single license term for the whole VNF.
  - In addition may include license terms for each of the VNF package artifacts if different from the one of the released VNF
- A directory for artifacts serving 3rd party extensions
  - Allow adding artifacts supporting external (non-MANO) extensions e.g. ONAP extensions

New feature in version 2.41

More information

- NFV Technology Page (information) **http://www.etsi.org/nfv**
- NFV Portal (working area) **http://portal.etsi.org/nfv**
- NFV Proofs of Concept (information) **http://www.etsi.org/nfv-poc**
- NFV Plugtest (information & registration) **http://www.etsi.org/nfvplugtest**
- Open Area:
  - Published Docs: https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf
  - Working Drafts http://docbox.etsi.org/ISG/NFV/Open/Drafts/
  - Issue tracker http://nfvwiki.etsi.org/index.php?title=NFV_Issue_Tracker
  - Detailed Release 3 specification progress can be found at: https://nfvwiki.etsi.org/index.php?title=Feature_Tracking

BACKUP

VNF Package (NFV SOL 004)

Items covered in ETSI GS NFV-SOL 004

- How to use CSAR
- Naming Conventions and Location for
  - Manifest file
  - Change History file
  - Testing files directory
  - Licensing information directory
  - Certificate files
- Naming Conventions for name-value pairs in the manifest file
- Security Features of the CSAR
  - Digests
  - Signature
  - Certificates
  - Encryption

VNF Package Structure (Option 1): TOSCA YAML CSAR with Metadata File

- The TOSCA.meta file includes block_0 with the Entry-Definitions keyword pointing to a TOSCA definitions YAML file used as entry for parsing the contents of the overall CSAR archive – MRF.yaml. Example:

  TOSCA-Meta-File-Version: 1.0

  CSAR-Version: 1.1

  Created-by: Company Name

  Entry-Definitions: Definitions/ MRF.yaml

- Any TOSCA definitions files besides the one denoted by the Entry-Definitions can be found by processing respective imports statements in the entry definitions file (or in recursively imported files)
- Any artifact files (e.g. scripts, binaries, configuration files) can be either declared explicitly through blocks in the TOSCA.meta file or pointed to by relative path names through artifact definitions in one of the TOSCA definitions files contained in the CSAR file.

```
!------TOSCA-Metadata
     !------TOSCA.meta
!------Definitions
     !----- MRF.yaml
     !----- OtherTemplates (e.g.,
            type definitions)
!------Files
     !----- ChangeLog.txt
     !----- MRF.cert
     !----- image(s)
     !----- other artifacts
     !------Tests
        !----- file(s)
     !------Licenses
        !----- file(s)
!------Scripts
     !----- install.sh
!----- MRF.mf
```

References:
- ETSI GS NFV-SOL 004
- TOSCA-Simple-Profile-YAML-v1.1

VNF Package Structure (Option 2): TOSCA YAML CSAR without Metadata File

- CSAR contains a single yaml (.yml or .yaml) file at the root of the archive – MRF.yaml
- The yaml file contains a metadata section with template_name and template_version metadata. This file is the CSAR Entry-Definition file
- The CSAR-Version is defined by the *template_version* metadata:

```
tosca_definitions_version:  tosca_simple_yaml_1_1
metadata:
  template_name: MRF
  template_author: Company Name
  template_version: 1.0
```

```
!--------- MRF.yaml

!--------- MRF.mf

!--------- MRF.cert

!--------- ChangeLog.txt

!--------- Tests
            !----- file(s)

!--------- Licenses
            !----- file(s)

!--------- Artifacts
            !----- install.sh
            !----- images
            !----- templates
            !----- start.yang
```

References:
- ETSI GS NFV-SOL 004
- TOSCA-Simple-Profile-YAML-v1.1

VNF Package
Manifest File with Optional security support

- VNF package **metadata**

- A list of blocks each is related to one file in the VNF package, including
  - **Source**: artifact URI
  - **Optional Algorithm**: name of an algorithm used to generate the hash
  - **Optional Hash**: text string corresponding to the hexadecimal representation of the hash

- **Optional Manifest file Signature**

metadata:
vnf_product_name: vMRF-1-0-0
vnf_provider_id: Acme
vnf_package_version: 1.0
vnf_release_data_time: 2017.01.01T10:00+03:00

Source: MRF.yaml
Algorithm: SHA-256
Hash: 09e5a788acb180162c51679ae4c998039fa6644505db2415e35107d1ee213943

Source: scripts/install.sh
Algorithm: SHA-256
Hash: d0e7828293355a07c2dccaaa765c80b507e60e6167067c950dc2e6b0da0dbd8b

Source: https://www.vendor_org.com/MRF/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 36f945953929812aca2701b114b068c71bd8c95ceb3609711428c26325649165

-----BEGIN CMS-----
MIGDBgsqhkiG9w0BCRABCaB0MHICAQAwDQYLKoZIhvcNAQkQAwgwXgYJKoZIhvcN
AQcBoFEET3icc87PK0nNK9ENqSxItVIoSa0o0S/ISczMs1ZIzkgsKk4tsQ0N1nUM
dvb05OXi5XLPLEtViMwvLVLwSE0sKlFIVHAqSk3MBkkBAJv0Fx0=
-----END CMS-----

References:
- IANA register for Hash Function Textual Names
  https://www.iana.org/assignments/hash-function-text-names/hash-function-text-names.xhtml

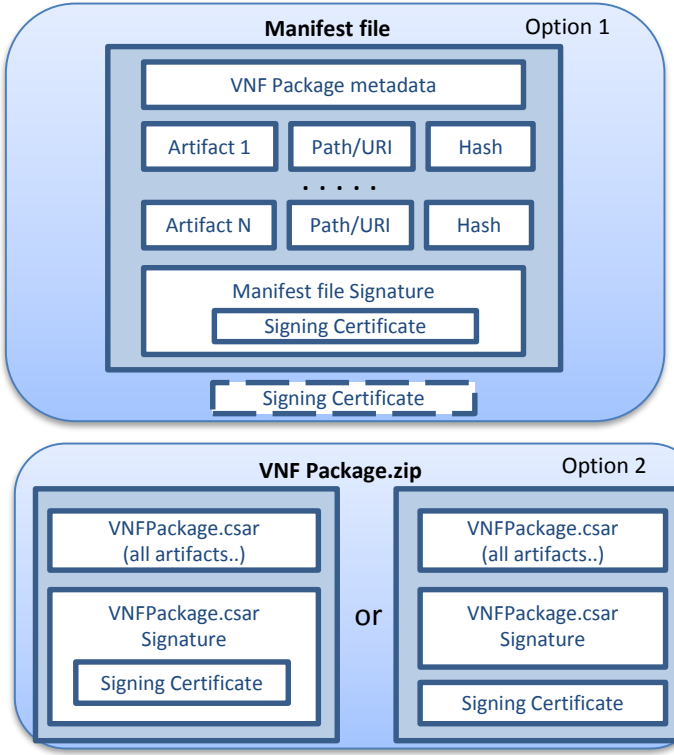
Adding Security to VNF Package Signing Individual Artifacts

- VNF provider may sign individual artifacts adding a signature file in standard format (e.g. CMS, PKCS#7)
- A certificate file with extension .cert accompany the signed artifact
- The signature and certificate files have the same name and location as the signed artifact
  - If the signature format allows it, the certificate may be included in the signature file

```
!--------- Artifacts
          !----- install.sh
          !----- images
                    !----- MRF.img
                    !----- MRF.cert
                    !----- MRF.cms
```

Adding Security to VNF Package
Public Key Based Integrity and Authenticity

- **Security Option 1**: **Manifest file** - based if there are both local and external artifacts
  - A Digest hash per each artifact
  - Manifest file is signed with VNF provider private key
  - VNF provider's certificate includes a VNF provider public key
  - The certificate may be a separate artifact or included in the signature container, e.g. CMS
- **Security Option 2**: **CSAR**-based if all artifacts are located inside a CSAR
  - CSAR file is digitally signed with the VNF provider private key
  - No digest hash per each artifact
  - VNF provider delivers one zip file containing a CSAR file, a signature file and a certificate file that includes a VNF provider public key
  - The certificate may be a separate artifact or included in the signature container, e.g. CMS
- Key different: Option 1 has two level security protection, whereas option 2 has one level
- Both options rely on existence in the NFVO of a root certificate of a trusted certificate authority, delivered via a trusted channel separately from a VNF package



Adding Security to VNF Package
Encrypting Security Sensitive Artifacts

- Asymmetric encryption:
  - VNF provider uses the user public key to encrypt the security sensitive artifact
  - A consumer of the artifact then decrypts the artifact with its own private key
- Symmetric encryption:
  - The artifact is encrypted with the VNF provider key shared with the consumer in encrypted form (a user key is used for encryption)
  - A consumer of the artifact decrypts the shared key with its own private key and then uses the obtained shared key to decrypt the artifact
  - The encrypted artifact is delivered in a CMS file, with all info needed to decrypt it: algorithm used for artifact encryption, encrypted key used for artifact encryption and algorithm used to encrypt the key.

A public key is provided by the party who is responsible to either on-board the package or use the artifact