

VNF Configuration

Understanding the available options

Presented by: **Bruno Chatras**

For: **SDN- NFV World Congress**

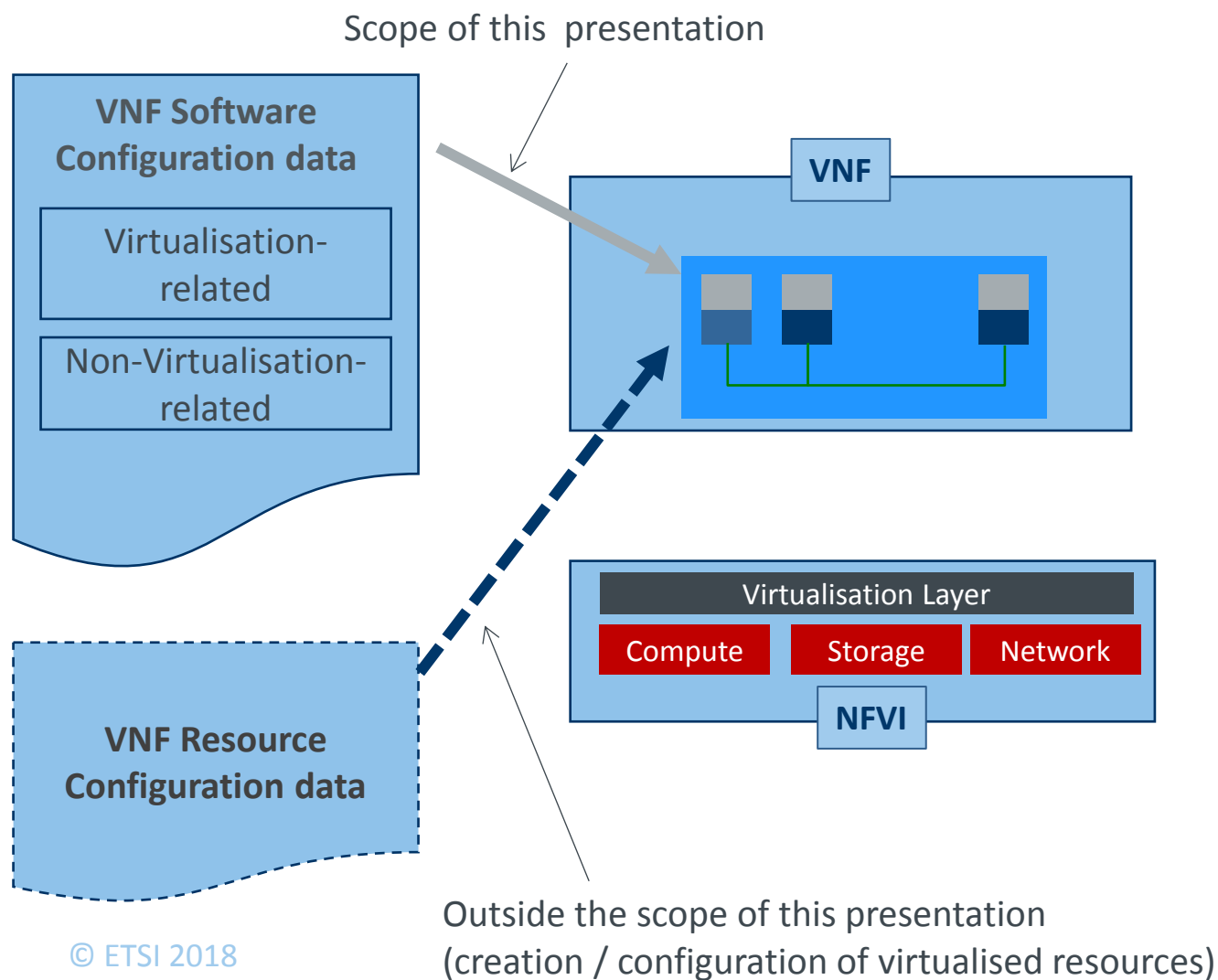
8 October 2018

Agenda

- ✓ Introduction
- ✓ Configuration Path #1
- ✓ Configuration Path #2
- ✓ Configuration Path #3
- ✓ Summary



Classification of VNF configuration data



Virtualisation-related configuration parameters

- ✔ Parameters whose value is or can be influenced by processing functions in the NFVI and/or NFV-MANO
- ✔ e.g. IP address of a Connection Point of a VNFC to be configured on another VNFC

Non-virtualisation-related configuration parameters

- ✔ Parameters whose value cannot be influenced by processing functions in the NFVI and/or MANO
- ✔ e.g. APN-to-GGSN mappings in a SGSN VNF

Alternative paths for VNF configuration

The ETSI NFV architectural framework enables 3 non-exclusive alternative paths to provide configuration data to a VNF software instance.

- ✓ Path #A: OSS -> (EM) -> VNF software instances
- ✓ Path #B: (OSS -> NFVO or EM ->) VNFM -> VNF software instances
- ✓ Path #C: (OSS -> NFVO or EM ->) VNFM -> VIM -> Virtualisation Container -> VNF software instances

OSS: Operations Support System
 EM: Element Manager
 VNFM: VNF Manager
 VIM: Virtualisation Infrastructure Manager



Configuration Path #A

CONFIGURATION Path #A

OSS -> (EM) -> VNF software instance

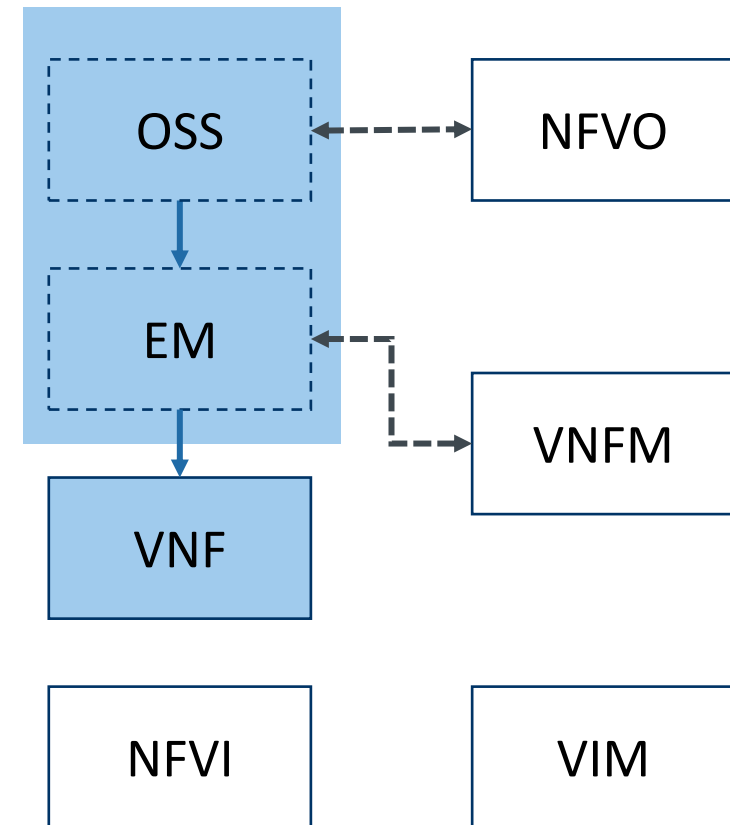
Relies on reference points currently out of the scope of the ETSI ISG NFV work: OSS-EM and/or EM-VNF.

Non-virtualisation-related configuration data

✔ Similar approach as for a Physical Network Function (PNF), with no MANO involvement.

Virtualisation-related configuration data

✔ Requires that the OSS gets the virtualisation-dependent values from the NFVO, through the NS LCM API defined in ETSI GS NFV-SOL 005 (e.g. use of the [Query NS operation](#)).



Configuration Path #B

CONFIGURATION PATH B: (OSS -> NFVO or EM ->) VNFM -> VNF software instance

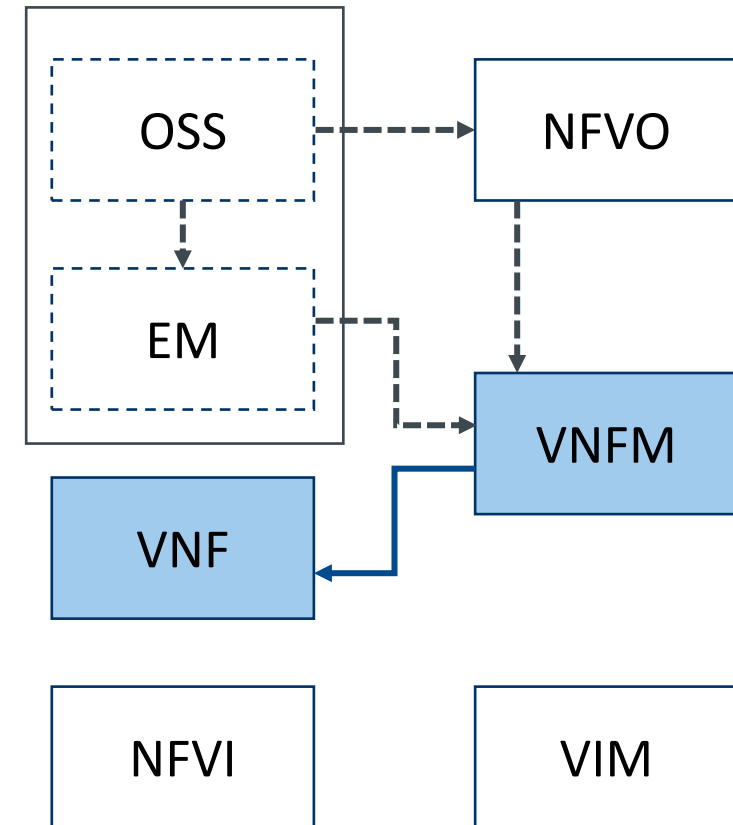
Relies on the support of the optional [VNF Configuration](#) interface (push mode) or on VNF LCM notifications followed by a [Query VNF](#) operation (Pull Mode) defined in ETSI GS NFV-IFA 008.

Non-virtualisation-related configuration data

- ✔ Relies on [the configurable properties](#) declared in the VNFD
- ✔ NFVO and VNFM are used as a “tunnelling” mechanism between the OSS and the VNF Application

Virtualisation-related configuration data

- ✔ Relies on the [configurable properties](#) declared in the VNFD and/or on pre-defined parameters available in [VnfInfo](#) (DHCP server address to use, Addresses and ports assigned to the Connection Points)



Optional Interactions
↔

Configurable properties

Configuration properties **must be declared in the VNFD:**

- ✔ at the VNF level: `VnfConfigurableProperties`
- ✔ and/or VDU level: `VnfcConfigurableProperties`

VNF configuration properties include

- ✔ Standard properties to enable/disable auto scaling and auto healing in a VNF instance.
- ✔ Additional configuration properties to be defined by VNF providers

VNFC configuration properties only include additional configuration properties to be defined by VNF providers

CONFIGURATION PATH B: PUSH and PULL Mode

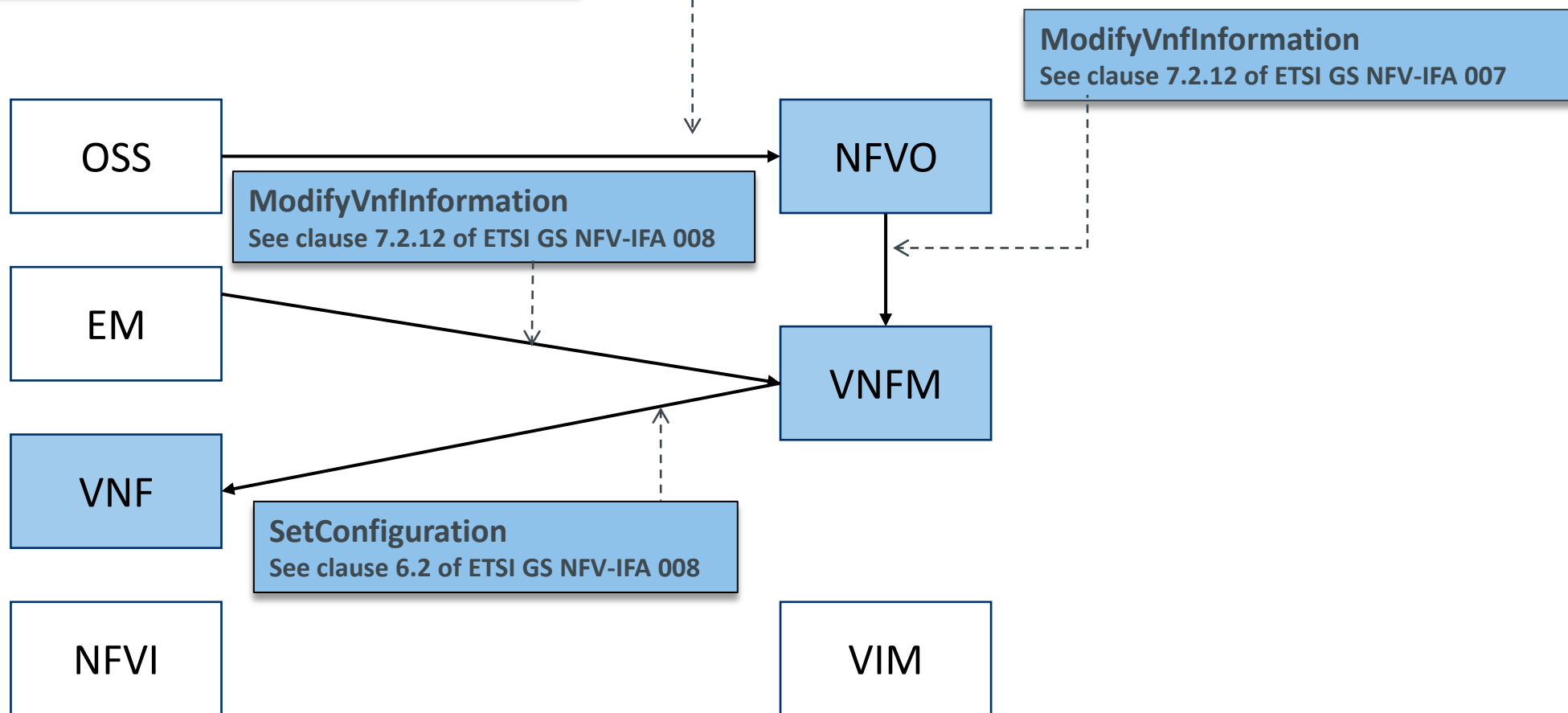
Configuration data can be provided to a VNF instance in push or pull mode

- ✔ The VNFM sends a [SetConfiguration](#) operation to the VNF instance as soon as a modification occurs of the [VNF configuration interface](#).
- ✔ At boot time or upon receipt of a notification that some information has changed, the VNF instance sends a [QueryVnf](#) operation to the VNFM.

Setting and transferring VNF configurable properties

UpdateNs with update type = **ModifyVnflInformation**
See clause 7.3.5 of ETSI GS NFV-IFA 013

*Pull mode on the VNF – VNFM reference point not shown



Configuration Path #C

CONFIGURATION PATH C

(OSS -> NFVO or EM) -> VNFM -> VIM -> NFVI -> VNF



Relies on the contents of the VNFD and the support by the NFVI and VIM of a mechanism to **push initial configuration data** to a virtualisation container (e.g. a VM)

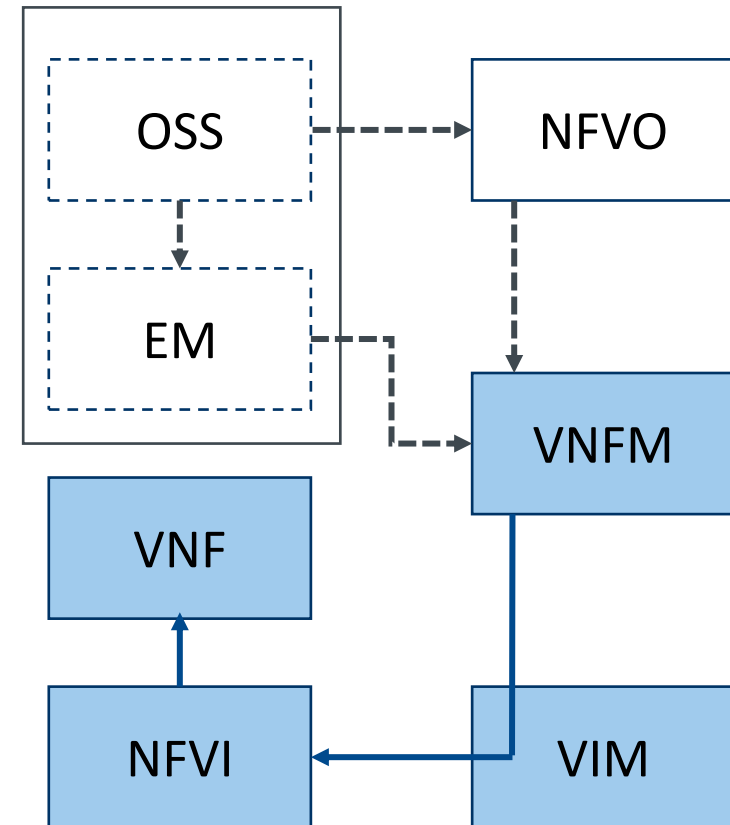
✓ e.g. Cloud_Init

Non-virtualisation-related configuration data

✓ Boot data available in the VNFD are pushed to the VNF instance via the NFVI. Might be customized with OSS-provided values.

Virtualisation-related configuration data

✓ Boot data available in the VNFD are customized by the VNFM and pushed to the VNF instance via the NFVI.



Optional Interactions
↔

The Boot data mechanism

Boot data is an attribute of a VDU (in the VNFD), representing **initialisation data** to be sent to VNFC instances, via the VIM and the virtualisation containers hosting them.

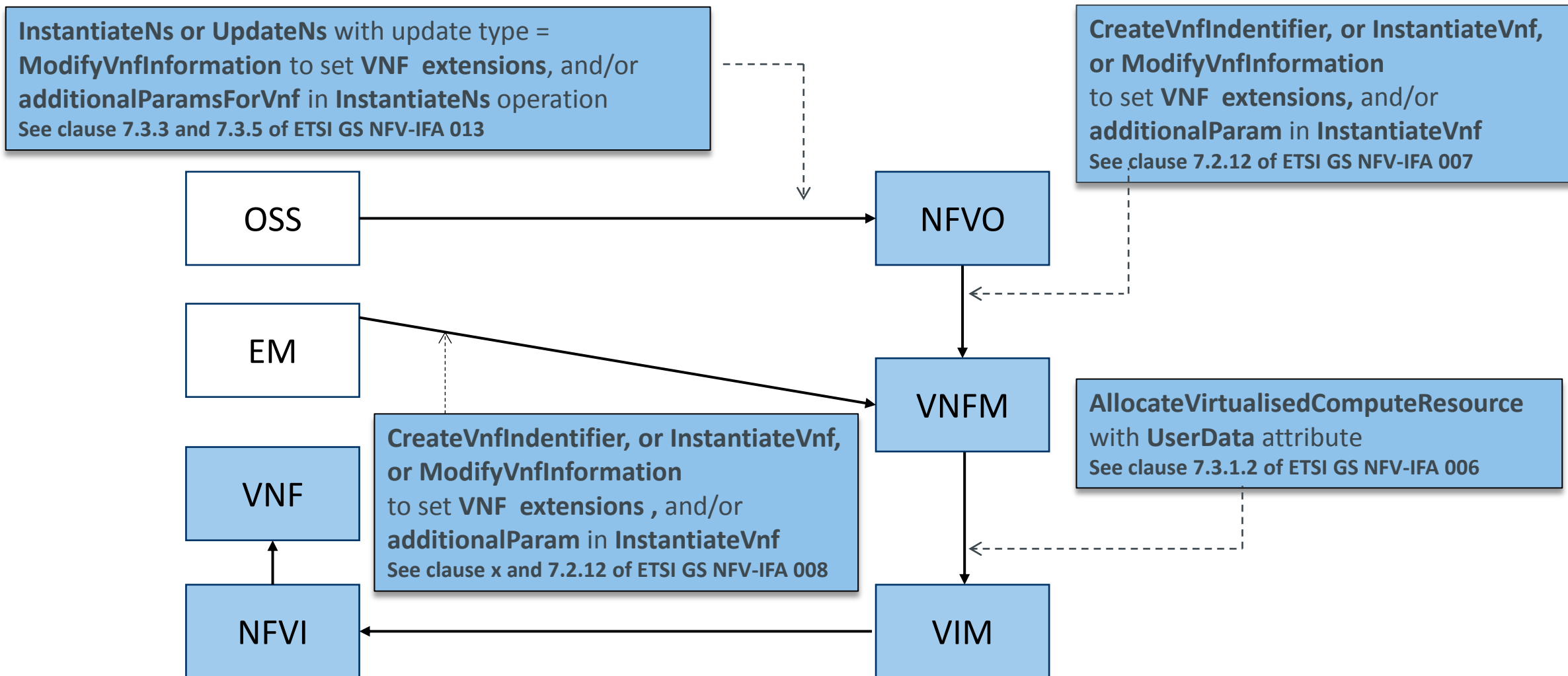
- ✓ Using the **AllocateVirtualisedComputeResource** operation defined in clause 7.3.1.2 of ETSI GS NFV-IFA 006

Can be a **string** or a **URL to an initialization file** contained in the VNF package.

May include volatile and/or persistent **variable parts** declared in other information of the VNFD

- ✓ **VnfLcmOperationsConfiguration** information element (see clause 7.1.5.2 of ETSI GS NFV-IFA 011) for volatile data available during the lifetime of a VNF lifecycle management operation, and/or
- ✓ **Extension** attribute of the **VnfInfoModifiableAttribute** information element (see clause 7.1.14 of ETSI GS NFV-IFA 011) for persistent data available during the lifetime of a VNF instance (i.e. set in **VnfInfo**).

Setting variable parts and transferring boot data to a VNF





Key Takeaways

In summary: The NFV architectural framework provides different options for configuring VNF instances



Path #A: OSS -> (EM) -> VNF software instances

- ✔ NFV-independent method, application configuration files may be included in the VNF Package
- ✔ Typical Use Case: Non-virtualisation related configuration, e.g. APN configuration in a PDN-GW

Path #B: (OSS -> NFVO ->) VNFM -> VNF software instances

- ✔ Supported by MANO operations and configurable properties declared in the VNFD.
- ✔ Typical Use Case: Configure a VNFC instance with information related to another VNFC instance (e.g. internal load balancer configuration)

Path #C: (OSS -> NFVO ->) VNFM -> VIM -> Virtualisation Container -> VNF software instances

- ✔ Supported by NFV-MANO operations and the “boot data” attribute declared in the VNFD. Communication with the VNF instances relies on NFVI/VIM mechanisms such as “Cloud_Init”.
- ✔ Typical Use Case: Root password, SSH keys, DNS server, custom script, etc.

ETSI NFV Specifications and Reports

ETSI NFV on etsi.org:

✓ <http://www.etsi.org/technologies-clusters/technologies/nfv>

Published deliverables:

✓ <http://www.etsi.org/technologies-clusters/technologies/nfv#specifications>

✓ <http://docbox.etsi.org/ISG/NFV/Open/Published/>

Draft deliverables (ongoing work):

✓ <http://docbox.etsi.org/ISG/NFV/Open/Drafts/>

API Specifications

https://nfvwiki.etsi.org/index.php?title=API_specifications#API_specifications

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.





Thank You!





Backup slides

Configurable Properties vs. Extensions and Metadata

Configurable Properties

VNF-specific configuration parameters to be set on VNF instances

Example: EM address (IFA011), name_prefix_in_vim, and dns_server (SOL001)

Extensions

VNF-specific attributes that affect the lifecycle management of VNF instances in the VNFM

Example: http_proxy (SOL001)

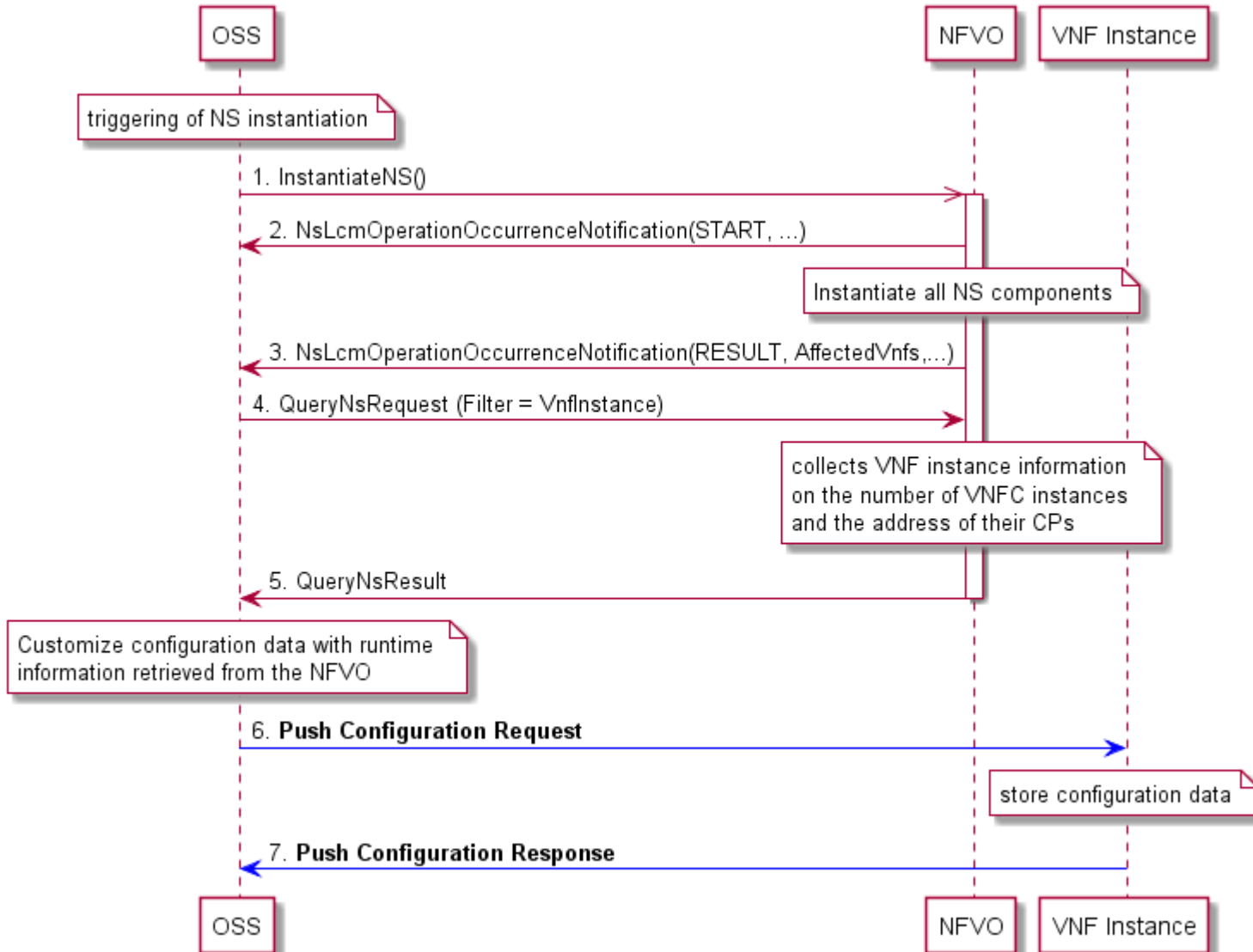
Metadata

VNF-specific metadata describing the VNF instance

Example: Tag in the VNFM a VNF instance as being going down soon

CONFIGURATION Path #A

Virtualisation-dependent configuration data example



The actual messages implementing the Push Configuration Request / Response information flow are outside the scope of ETSI NFV standardization.

The configuration data model can – but not need to - rely on application configuration templates embedded in the VNF Package (as non-MANO artefacts) along with – but not referenced from - the VNFD.

Configurable properties

Configurable properties represent **data to be configured** on a VNF instance.

Property values can be set

- ✔ In the VNFD (default values)
- ✔ By lifecycle management (LCM) scripts referenced in the VNFD
- ✔ The NFVO or the EM using the **Modify VNF Information** operation
 - ✔ See ETSI GS NFV-IFA 008 Clause 9.2.2.2

TOSCA representation of VNF configurable properties

```

tosca_definitions_version: toska_simple_yaml_1_2

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

```

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

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

- ✎ The TOSCA representation of a VNFD is specified in Draft [ETSI GS NFV-SOL001](#)
A YANG representation is also available in Draft [ETSI GS NFV-SOL 006](#)

VNF Configuration interface

✔ This interface allows the VNFM to set the configuration of a VNF instance and/or its VNFC instance(s).

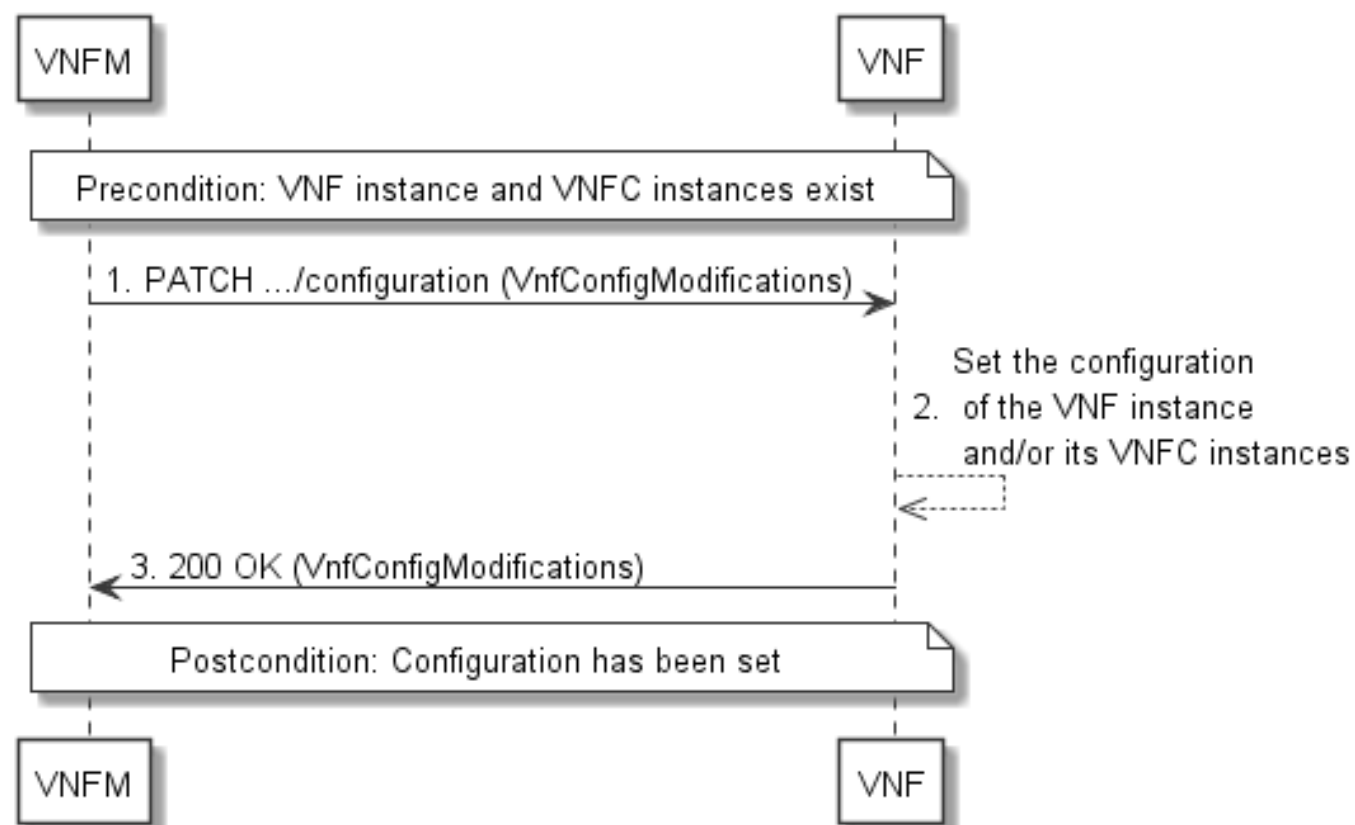
- ✔ Configurable properties
- ✔ DHCP server address
- ✔ Addresses and ports assigned to the Connection Points

✔ It contains a single operation: SetConfiguration

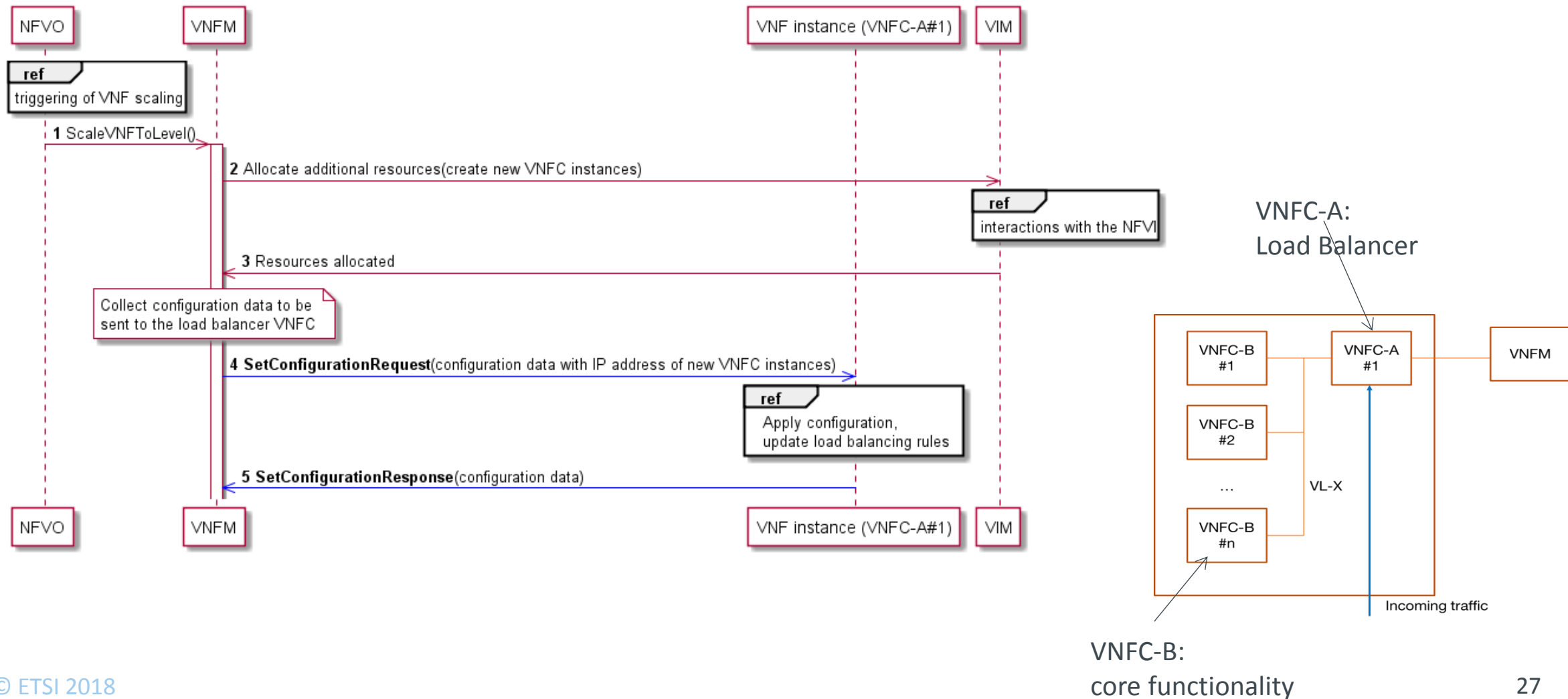
✔ This operation is implemented using an HTTP PATCH method.

✔ The support of the VNF configuration interface is optional.

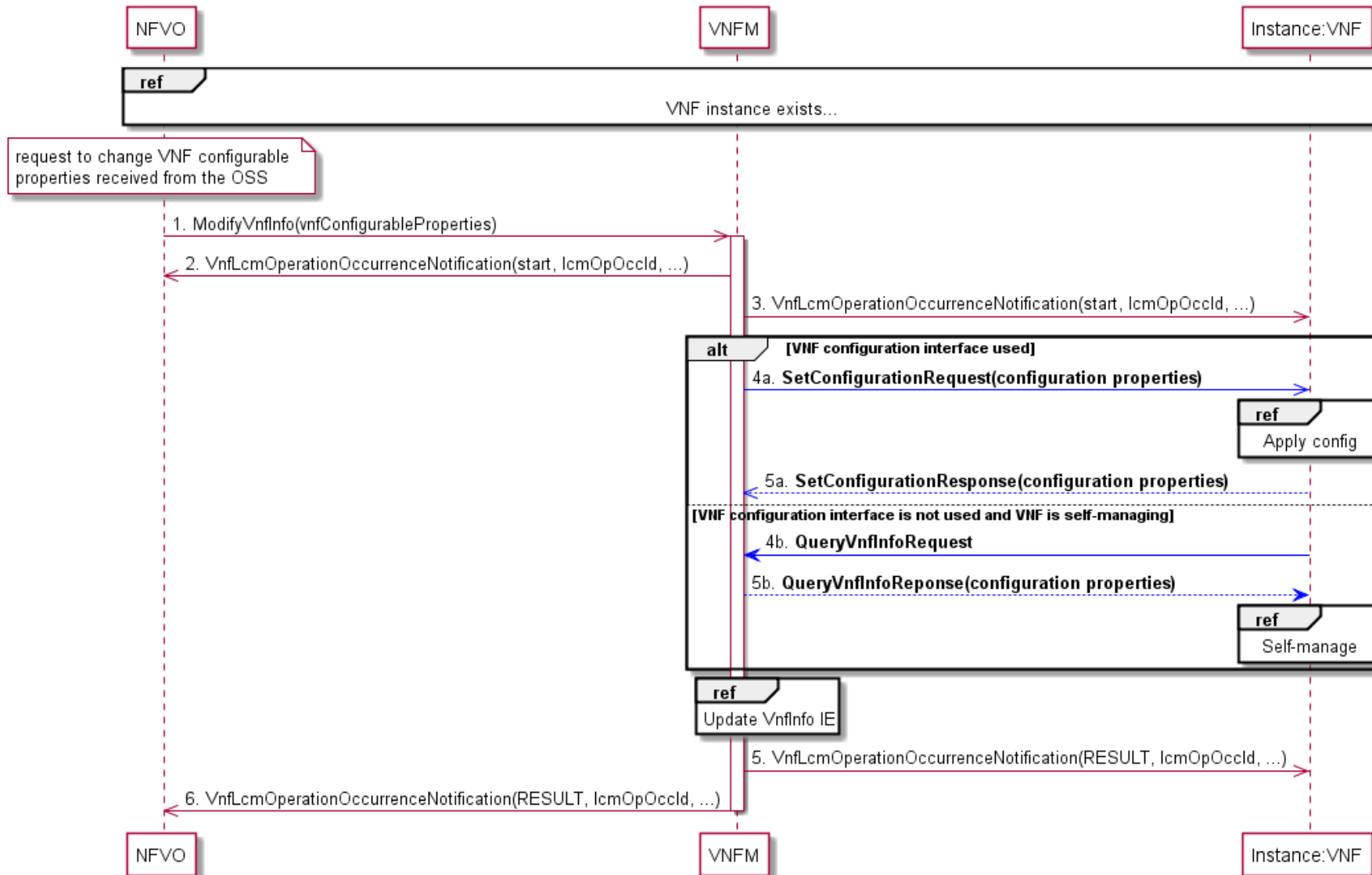
See also clause 6.2 of [ETSI GS NFV-IFA 008](#) (Stage 2) and Clause 9 of [ETSI GS NFV-SOL 002](#) (Stage 3).



CONFIGURATION PATH B: Example with standard properties



CONFIGURATION PATH B: Example with VNF Configurable Properties

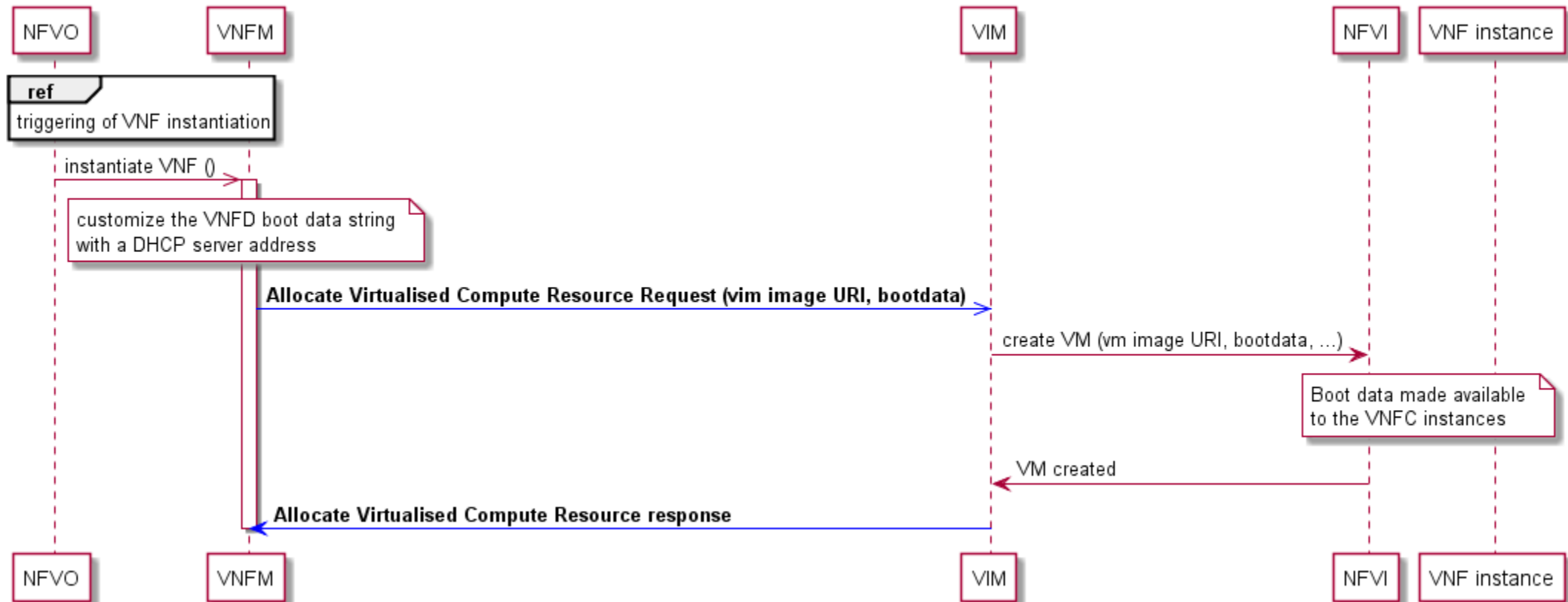


Simplified

(Not all information flows are shown)

CONFIGURATION PATH C

Example: DHCP server address configuration



TOSCA representation of boot data in a VNFD

```

tosca_definitions_version: toska_simple_yaml_1_2
..
node_types:
  mycompany.nodes.nfv.SunshineDB.1_0.1_0:
    derived_from: toska.nodes.nfv.VNF
    properties:
      ..
      modifiable_attributes:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributes
      ..
data_types:
  mycompany.datatypes.nfv.VnfInfoModifiableAttributes:
    derived_from: toska.datatypes.nfv.VnfInfoModifiableAttributes
    properties:
      extensions:
        type: mycompany.datatypes.nfv.VnfInfoModifiableAttributesExtensions
        required: false

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

```

```

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

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

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

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

```