Welcome to the World of Standards

ETSI

World Class Standards

VNF PACKAGE SPECIFICATION GS NFV-SOL004

Bruno Chatras, ETSI NFV SOL Chair, Orange

Thinh Nguyenphu, ETSI NFV SOL Vice-chair, Nokia

Andrei Kojukhov, VNF Package Spec Rapporteur, Amdocs

PART 1: ETSI NFV SOL ACTIVITIES

ETSI NFV ISG Solutions (SOL) Working Group

Feb’13	ETSI NFV ISG Created
Dec’13	ETSI NFV Architectural Framework v1.2.1 Published
Nov’14	**Release 2 work starts**
Apr’16	Release 2 work on APIs, protocols and data models starts
	Release 3 work starts
Sep’16	Completion of Release 2 work on requirements, interfaces and information model
Now	First set of API specifications and VNF package specification published
2017Q4	Remaining API specifications and NFV descriptors specifications published

SOL WG Scope

- **Protocols and APIs**
- **Data Models**

SOL Work Items

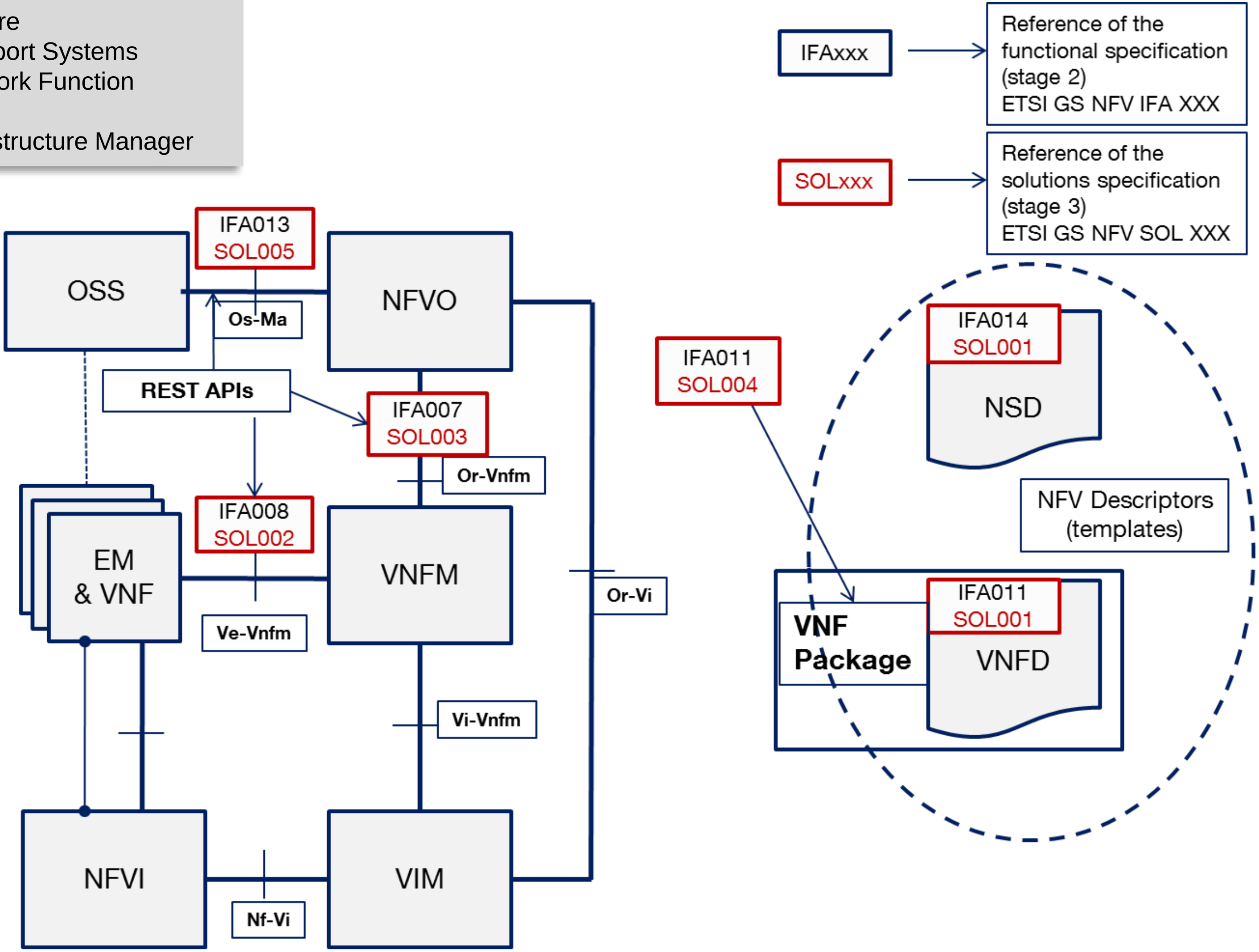
- **SOL001 TOSCA-based VNFD and NSD specification**
- **SOL002 REST APIs for the Ve-Vnfm reference point**
- **SOL003 REST APIs for the Or-Vnfm reference point**
- **SOL004 TOSCA-based VNF Package specification**
- **SOL005 REST APIs for the Os-Ma reference point**

Our goal

- **Enable an open ecosystem where VNFs and Network Services are interoperable with independently developed NFV management and orchestration systems and where the components of these systems are themselves interoperable.**

Overview of SOL specifications

EM = Element Manager
NFVO = NFV Orchestrator
NFVI = NFV Infrastructure
OSS = Operations Support Systems
VNF = Virtualised Network Function
VNFM = VNF Manager
VIM = Virtualised Infrastructure Manager



WHERE WE ARE

Completed work

- Specification of a set of REST APIs applicable to the VNFM-NFVO, and VNFM-VNF/EM reference points.
- Specification of a VNF Packaging format based on TOSCA CSAR.

Ongoing work (to be completed by the end of 2017)

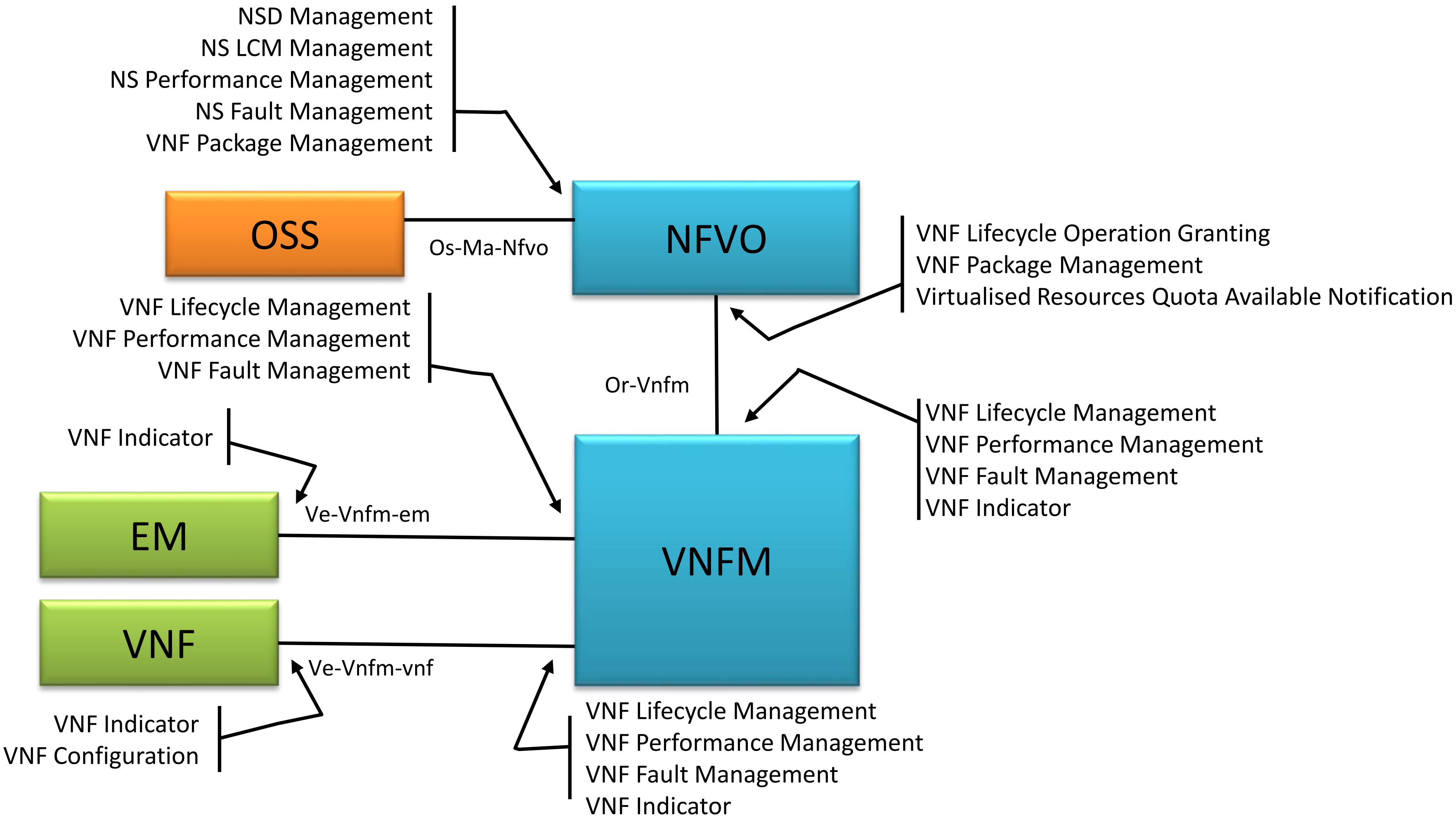
- Specification of a set of REST APIs applicable to the OSS-NFVO reference point.
- Specification of a TOSCA profile for the VNFD and NSD.

Next Steps

- ETSI will provide an on-line repository with a Swagger (a.k.a. OneAPI) representation of the APIs, with associated tools (e.g. Swagger UI, Bug Tracking).
- Maintenance, Bug fixing, based on industry feedback
- Conformance testing specifications for REST APIs

Final publications: http://www.etsi.org/nfv (Specifications tab)
Draft specifications :http://docbox.etsi.org/ISG/NFV/Open/Drafts/
Bug reports: http://nfvwiki.etsi.org/index.php?title=NFV_Issue_Tracker

RESTful APIs developed by the SOL WG



The figure represents the current snapshot of ETSI NFV specifications.

REST in ETSI NFV SOL

- Use basic HTTP methods implementing CRUD(*) functionality
  - POST – create resource
  - GET – read / query resource(s)
  - PATCH – update resource
  - DELETE – delete resource

- on a set of simple resources:
  - VNF instances
  - VNF lifecycle management operation occurrences
  - Subscriptions
  - Notification endpoints
  - PM Jobs
  - PM Thresholds
  - Alarms
  - VNF Indicators
  - Grants
  - VNF Packages
  - …

Past Webinar on
BrightTALK
Common APIs for NFV Interop

- Use TASK resources to expose complex operations (e.g. Scale VNF)

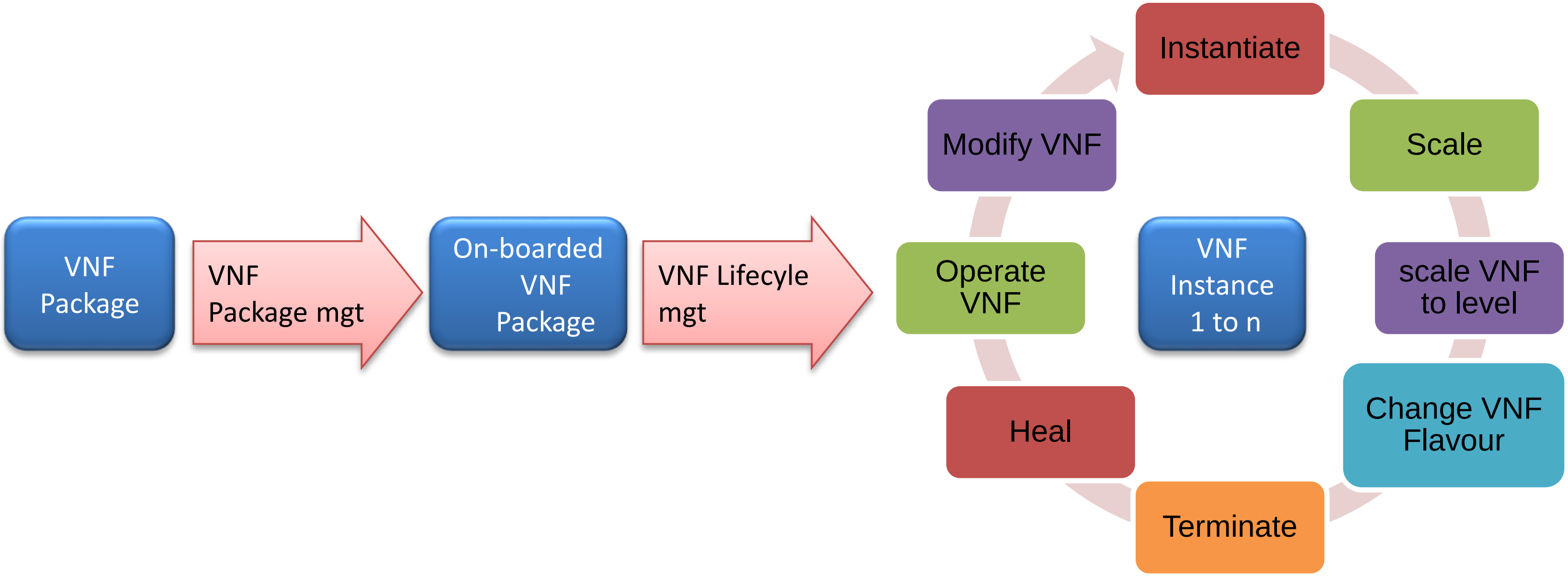
- Use JSON to encode HTTP requests and responses bodies.

(*) CRUD = Create, Read, Update, Delete

PART 2: THE VNF PACKAGE SPECIFICATION

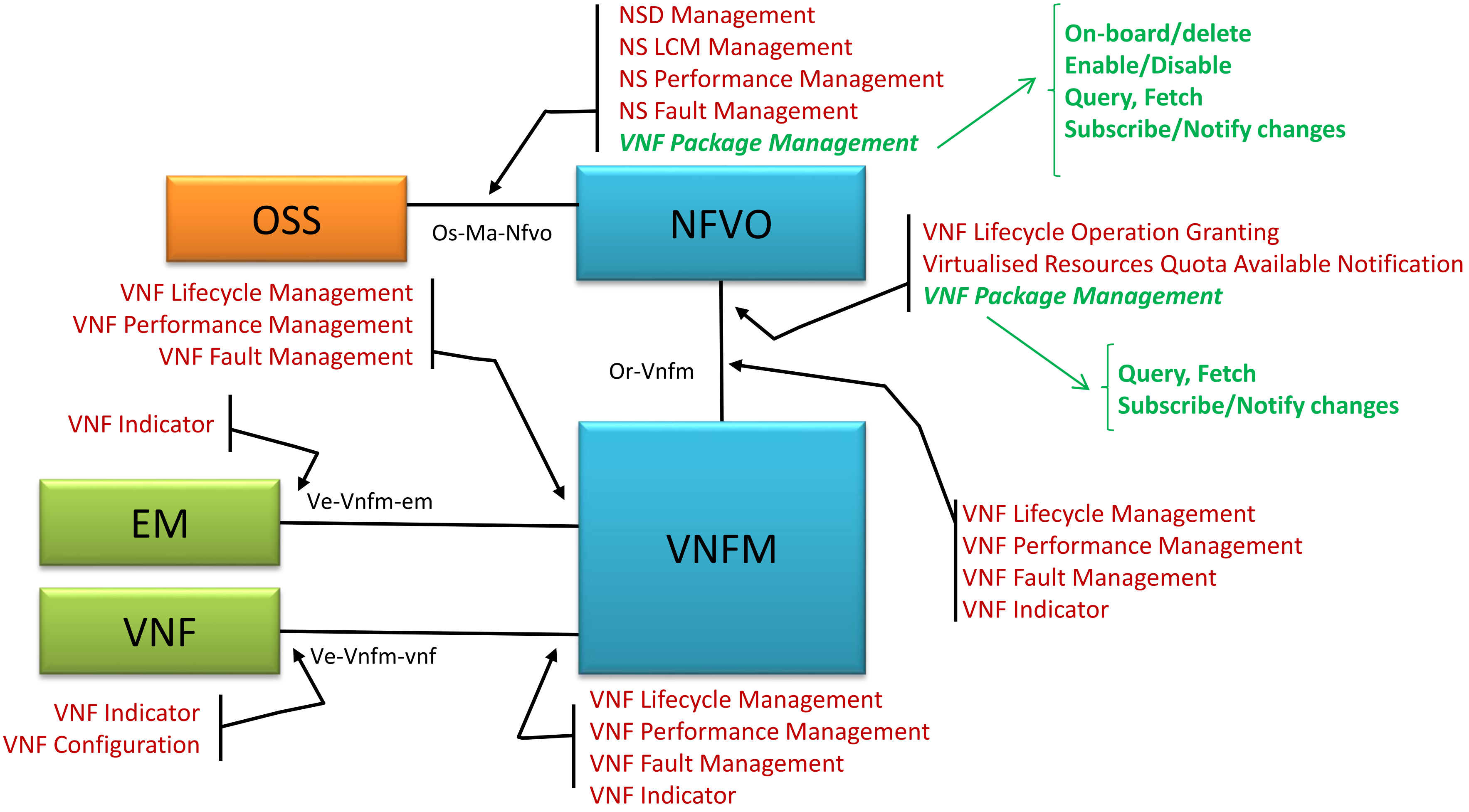
VNF Package Onboarding

- On-boarding a VNF package is a pre-requisite to VNF instantiation and lifecycle management



Note: see next slide for detail of all operations

VNF Package Management



The figure represents the current snapshot of ETSI NFV specifications.

Use Case: VNF Package Management: On-Boarding

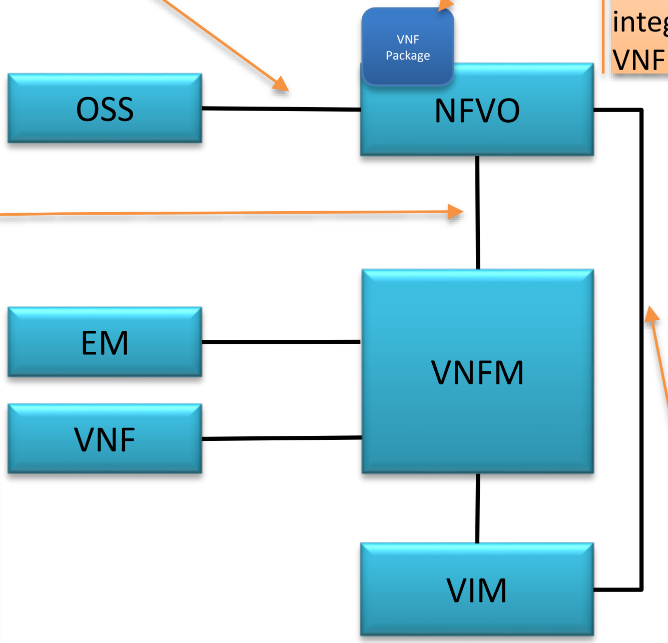
1) VNF Package is onboarded by the OSS into the NFVO, which then manages the on-boarded VNF Packages

1a) During the on boarding of the VNF Package, a validation of the package is performed. The validation is a procedure that verifies the integrity and authenticity of the VNF Package.

3) All subscribers (including VNFM) get notified when VNF packages are onboarded or removed

4) The VNFM can obtain the VNFD and information about the VNF Package from the NFVO by performing a query.

5) VNFM fetches either a whole VNF Package or selected artifacts



2) The NFVO could prepare the VIM(s) for instantiation of the onboarded VNFs, by downloading software images to the VIMs

1) Onboard VNF package Operation:
   - OnboardVnfPackageRequest/Response
2) Add Image Operation:
   - AddImageRequest/Response
3) Subscription & Notify Operation:
   - SubscribeRequest/Response
   - NotifyRequest/Response
4) Query On-boarded VNF Package Info Operation:
   - QueryOnboardedVnfPkgInfoRequest/Response
5) Fetch VNF Package Operation:
   - FetchVnfPackageRequest/Response

Note: Rel. 2 specification do not specify the sequence of these operations. This use case example is only for illustration.

Reference:
- ETSI GS NFV-IFA 013
- ETSI GS NFV-IFA 005
- ETSI GS NFV-SOL 005

Why VNF Package Standard Is Needed?

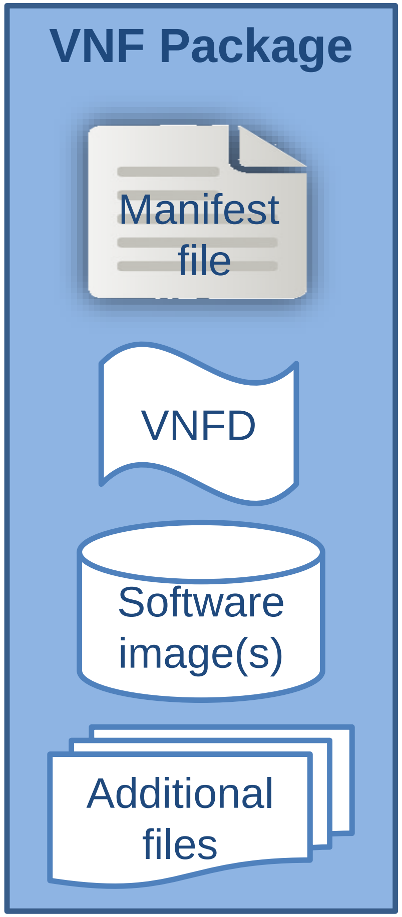
- There is a need for a uniform way for VNF providers to deliver VNFs to service providers and making the delivery simple, effective and efficient

- TOSCA YAML CSAR 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

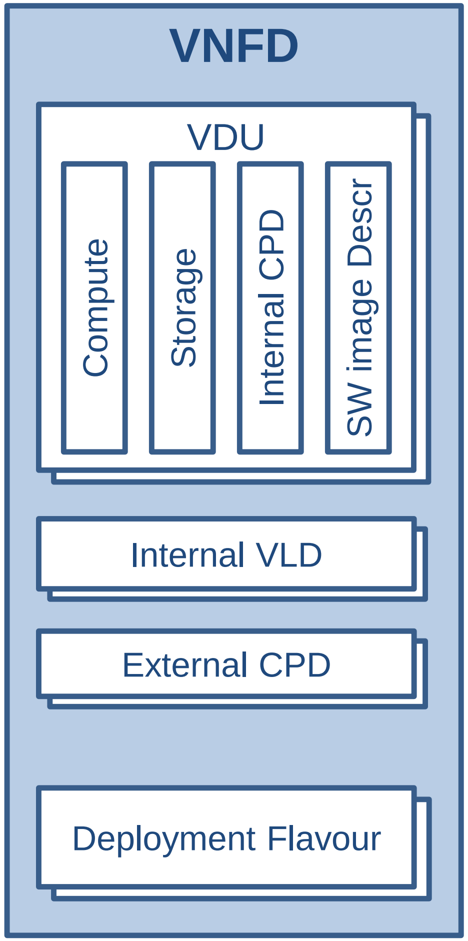
- The VNF Package contains:
  - the VNF descriptor (VNFD) that defines metadata for package onboarding and VNF management,
  - the software images needed to run the VNF, and
  - Manifest file that provides package integrity and authenticity
  - (optional) additional files to manage the VNF (e.g. scripts, 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 Structure: VNFD

- The VNFD defines VNF properties and requirements, such as:
  - Resources needed (amount and type of Virtual Compute, Storage, Networking),
  - Connectivity:
    - External Connection Points (described via CP Descriptors, CPD).
    - Internal Virtual Links (described via VL Descriptors, VLD)
    - Internal Connection Points (described via CP Descriptors, CPD)
  - LCM behavior (e.g. scaling, instantiation), operations, and configuration
  - References to SW images, LCM scripts and other files located or referred in VNF package
  - Affinity / anti-affinity and other policy rules
  - Deployment flavours (size-bounded deployment configurations, e.g. related to capacity).



- The VNFD is the main input to VNF instances lifecycle management

References:
- ETSI GS NFV-IFA 011
- ETSI GS NFV-SOL 001

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 Specification of the CSAR format.

- The VNFD is the main TOSCA definitions YAML file inside the archive.

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

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

  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
Standard Artifacts and Directories

- Change History File
  - Humanly readable text file
  - All the changes in the VNF package shall be versioned, tracked and inventoried

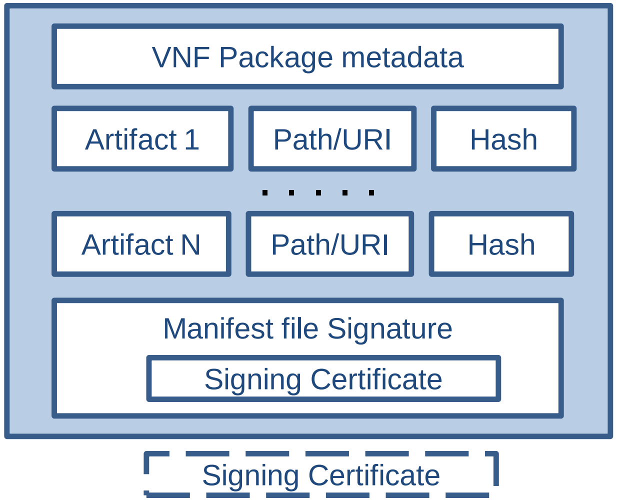
- Testing Files
  - Goal is to enable VNF package validation
  - VNF Provider includes files containing necessary information (e.g. test description)

- 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

Adding Security to VNF Package
Public Key Based Integrity and Authenticity

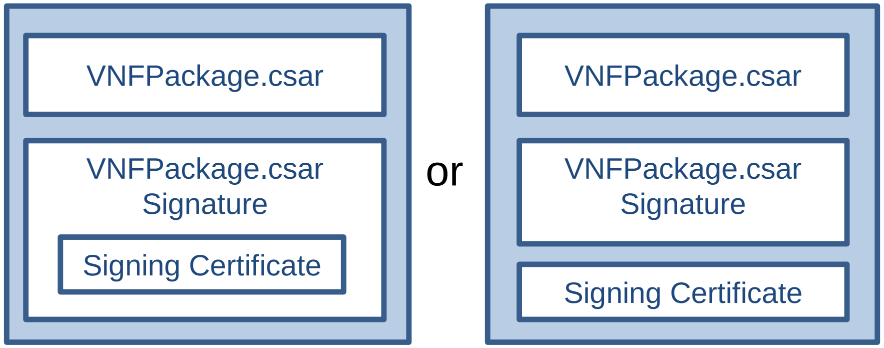
- Option 1: **Manifest file** - based if there are both local and external artifacts
  - A Digest hash per each artifact
  - Manifest file is signed with VNF provider private key
  - VNF provider's certificate includes a VNF provider public key
  - The certificate may be a separate artifact or included in the signature container, e.g. CMS

Manifest file



- Option 2: **CSAR**-based if all artifacts are located inside a CSAR
  - CSAR file is digitally signed with the VNF provider private key
  - 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

VNF Package.zip



- 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

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

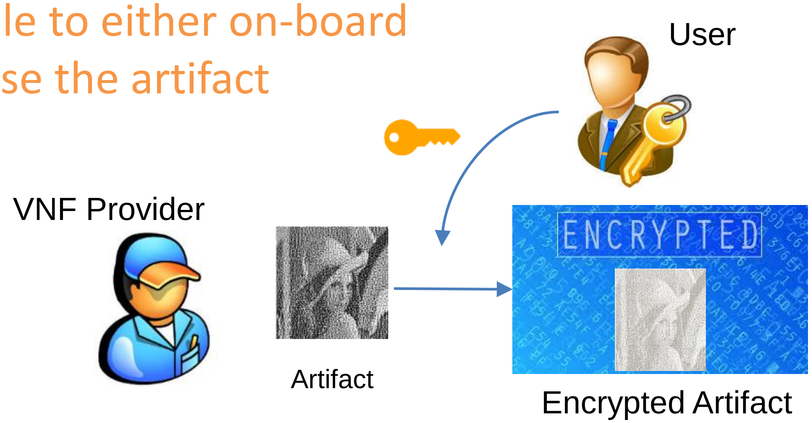
- The signature and certificate files have the same name and location as the signed artifact
  - If the signature format allows it, the certificate may be included in the signature file

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

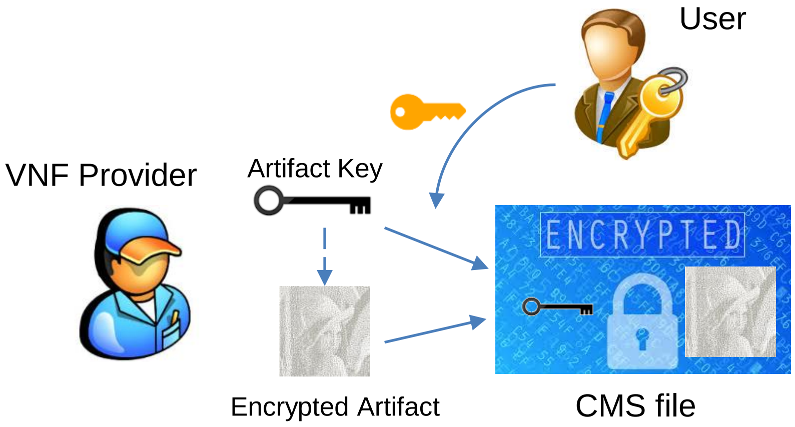
Adding Security to VNF Package
Encrypting Security Sensitive Artifacts

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

- 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.



World Class Standards

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:

Drafts http://docbox.etsi.org/ISG/NFV/Open/Drafts/

Issue tracker http://nfvwiki.etsi.org/index.php?title=NFV_Issue_Tracker