ETSI GS NFV-SOL 004 V4.3.1 (2022-07)

Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; VNF Package and PNFD Archive specification

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/NFV-SOL004ed431

Keywords
data, NFV, protocol, virtualisation

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

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 the structure and format of a VNF package file and its constituents, fulfilling the requirements specified in ETSI GS NFV-IFA 011 [1] for a VNF package.

The present document also specifies the structure and format of a PNFD archive file and its constituents, fulfilling the requirements specified in ETSI GS NFV-IFA 014 [i.9] for a PNFD archive.

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.

[1] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification".

[2] OASIS TOSCA-Simple-Profile-YAML-v1.1-csprd01: "TOSCA Simple Profile in YAML Version 1.1".

NOTE: Available at http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/csprd01/TOSCA-Simple-Profile-YAML-v1.1-csprd01.html.

[3] IETF RFC 3339: "Date and Time on the Internet: Timestamps".

[4] IANA register for Hash Function Textual Names.

NOTE: Available at https://www.iana.org/assignments/hash-function-text-names/hash-function-text-names.xhtml.

[5] IETF RFC 5652 (September 2009): "Cryptographic Message Syntax (CMS)".

[6] IETF RFC 7468: "Textual Encodings of PKIX, PKCS, and CMS Structures".

[7] Void.

[8] Recommendation ITU-T X.509: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

[9] Void.

[10] IETF RFC 2315: "PKCS #7: Cryptographic Message Syntax Version 1.5".

[11] OASIS TOSCA-Simple-Profile-yaml-v1.3: "TOSCA Simple Profile in YAML Version 1.3".

NOTE: Available at https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.html.

[12] ISO/IEC 21320-1: "Information technology -- Document Container File -- Part 1: Core".

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] Void.

[i.2] Void.

[i.3] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.4] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[i.5] ETSI NFV registry of non-MANO artifact sets.

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

[i.6] ETSI GS NFV-SOL 006: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG specification".

[i.7] ETSI GS NFV-SOL 004 (V2.4.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".

[i.8] ETSI GS NFV-SOL 004 (V2.5.1): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".

[i.9] ETSI GS NFV-IFA 014: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification".

[i.10] ETSI GS NFV-SOL 005: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

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.3] and the following apply:

non-MANO artifact: artifact for use by functional blocks beyond NFV-MANO

non-MANO artifact set: set of related non-MANO artifacts which are intended to be used together

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASCII	American Standard Code for Information Interchange
CA	Certificate Authority
CMS	Cryptographic Message Syntax
CSAR	Cloud Service ARchive
IANA	Internet Assigned Number Association
MANO	Management and Orchestration
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
PKCS	Public Key Cryptographic Standard
PNF	Physical Network Function
PNFD	PNF Descriptor
TOSCA	Topology and Orchestration Specification for Cloud Applications
URI	Universal Resource Identifier
UTF	Unicode Transformation Format
VIM	Virtual Infrastructure Manager
VNF	Virtualised Network Function
VNFD	VNF Descriptor
YAML	YAML Ain't Markup Language
YANG	Yet Another Next Generation

4 VNF package

4.1 TOSCA YAML Cloud Service Archive (CSAR) overview

4.1.1 CSAR structure

TOSCA YAML CSAR file is an archive file using the ZIP file format whose structure complies with the TOSCA Simple Profile YAML v1.1 [2] or the TOSCA Simple Profile in YAML v1.3 [11]. According to the TOSCA Simple Profile YAML v1.1 [2], the CSAR file shall have one of the two following structures:

- CSAR containing a *TOSCA-Metadata* directory, which includes the TOSCA.meta metadata file providing an entry information for processing a CSAR file.
- CSAR without a *TOSCA-Metadata* directory and containing a single yaml file with a .yml or .yaml extension at the root of the archive. The yaml file is a TOSCA definition template that shall contain a metadata section with *template_name* and *template_version* keyname.

In addition, the CSAR file may optionally contain other directories with bespoke names and contents.

4.1.2 CSAR with TOSCA-Metadata directory

4.1.2.1 General

The TOSCA.meta metadata file includes *block_0* with the *Entry-Definitions* keyword pointing to a TOSCA definitions YAML file and optionally the Other-Definitions keyword as specified in TOSCA Simple Profile YAML v1.3 [11] pointing to other TOSCA definitions YAML files used as entries for parsing the contents of the overall CSAR archive.

Any TOSCA definitions files besides the one denoted by the *Entry-Definitions* and Other-Definitions keyword can be found by processing respective *imports* statements in the entry definitions files (or in recursively imported files).

Any additional artifacts 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 as described in TOSCA Simple Profile YAML v1.1 [2].

Extension of the TOSCA.meta file is described in clause 4.1.2.2.

In order to indicate that the simplified structure (i.e. not all files need to be declared explicitly) of TOSCA.meta file allowed by TOSCA Simple profile YAML 1.1 [2] is used, the *CSAR-Version* keyword listed in block_0 of the meta-file denotes the version 1.1 as described in the below example.

EXAMPLE:

```
TOSCA-Meta-File-Version: 1.0
CSAR-Version: 1.1
Created-by: Onboarding portal
Entry-Definitions: Definitions/MainServiceTemplate.yaml
```

END OF EXAMPLE.

4.1.2.2 TOSCA.meta file extension

The TOSCA.meta file structure extension is used when files defined in clauses 4.3.2 to 4.3.6 of the present document are included in the VNF package and when using CSAR with TOSCA-Metadata directory, as described in clause 4.1.2.1.

NOTE: TOSCA Simple Profile YAML v1.1 [2] does not preclude the TOSCA.meta file block _0 to be extended with key value pairs.

4.1.2.3 TOSCA.meta file keynames extension

Table 4.1.2.3-1 specifies an extension of the list of recognized TOSCA.meta file keynames as specified in the present document for the TOSCA.meta file. The keynames represents the entries for artifacts defined in clauses 4.3.2 to 4.3.6 of the present document and shall be located in the block_0.

Table 4.1.2.3-1: List of TOSCA-meta file keynames extensions

Keyname	Required	Type	Description
ETSI-Entry-Manifest	yes	string	Location of the Manifest file as defined in clause 4.3.2
ETSI-Entry-Change-Log	yes	string	Location of the Change history file as defined in clause 4.3.3
ETSI-Entry-Tests	no	string	Location of the Testing files as defined in clause 4.3.4
ETSI-Entry-Licenses	yes	string	Location of the Licensing information as defined in clause 4.3.5
ETSI-Entry-Certificate	no	string	Location of the Certificate file as defined in clause 4.3.6

Use of the Entry-Manifest, Entry-Change-Log, Entry-Tests, Entry-Licenses and Entry-Certificate keynames defined in ETSI GS NFV-SOL 004 versions 2.4.1 [i.7] to 2.5.1 [i.8] of the present document is deprecated. These keynames are only provided for backward compatibility with legacy VNF Package consumers; VNF package providers are warned that support of these keynames can be removed in subsequent versions of the present document. The key with and without the ETSI-prefix should not be both present in the TOSCA.meta. If both are present they shall point to the same value.

EXAMPLE:

```
TOSCA-Meta-File-Version: 1.0
CSAR-Version: 1.1
Created-By: MyCompany
Entry-Definitions: MRF.yaml
ETSI-Entry-Manifest: MRF.mf
ETSI-Entry-Licenses: Files/Licenses
ETSI-Entry-Change-Log: Files/ChangeLog.txt
```

END OF EXAMPLE.

4.1.3 CSAR without TOSCA-Metadata directory

4.1.3.1 General

This CSAR structure is only applicable if a YANG-based VNFD as defined in ETSI GS NFV-SOL 006 [i.6] or a TOSCA-based VNFD with single deployment flavour design as defined in clause 6.11.3 in ETSI GS NFV-SOL 001 [i.4] is included in the VNF Package. The yaml file at the root of the archive is the *CSAR Entry-Definition* file. The CSAR-Version is defined by the *template_version* metadata as can be seen in the below example. The value of *template_version* shall be set to 1.1.

EXAMPLE:

```
tosca_definitions_version: tosca_simple_yaml_1_2
metadata:
   template_name: MainServiceTemplate
   template_author: Onboarding portal
   template_version: 1.1
```

END OF EXAMPLE.

4.1.3.2 TOSCA Entry definition file metadata extension for a YANG-based VNFD

Table 4.1.3.2-1 specifies an extension of the list of recognized metadata keynames as specified in TOSCA-Simple-Profile-YAML-v1.1 [2] for the main TOSCA Service Template.

Table 4.1.3.2-1: List of metadata keynames extensions

Keyname	Required	Type	Description
yang_definitions	no	string	Reference to a YANG definition file representing the VNFD within a VNF Package

If a YANG-based VNFD is included in the VNF Package, the main TOSCA definitions YAML file shall include a metadata section with an additional metadata entry, where the keyname is "yang_definitions" and the value is the path to the YANG file representing the VNFD within the VNF Package. No additional contents shall be included in the main TOSCA definitions YAML file.

NOTE: The above requirement ensures that there cannot be both a YANG-based and a TOSCA-based representation of a VNFD in the same package.

EXAMPLE:

```
tosca_definitions_version: tosca_simple_yaml_1_1
metadata:
template_name: MainServiceTemplate
template_author: Onboarding portal
template_version: 1.1
yang_definitions: Definitions/myvnfd.xml
```

END OF EXAMPLE.

4.1.4 Void

4.2 VNF package structure and format

The structure and format of a VNF package shall conform to the TOSCA Simple Profile YAML v1.1 Specification of the CSAR format [2]. The zip file format shall conform to Document Container Format File [12].

NOTE: This implies that the VNF package can be structured according to any of the two options described in clause 4.1.

The consumer of a VNF package complying with the present document shall be able to process a CSAR file structured according to any of the two options described in clause 4.1. If the CSAR file contains a TOSCA-Metadata directory and a single yaml file with a .yml or .yaml extension at the root of the archive, the TOSCA.meta file contained in the TOSCA-Metadata directory shall be used as an entry information for processing the CSAR file.

4.3 VNF package file contents

4.3.1 General

A VNF Package shall contain a main TOSCA definitions YAML file representing all or part of the VNFD, and additional files. It shall be structured according to one of the CSAR structure options described in clause 4.1.

NOTE 1: ETSI GS NFV-SOL 001 [i.4] specifies the structure and format of the VNFD based on TOSCA specifications.

NOTE 2: ETSI GS NFV-SOL 006 [i.6] specifies the structure and format of the VNFD based on YANG specifications.

Examples of VNF package options are described in annex A.

4.3.2 VNF package manifest file

A CSAR VNF package shall have a manifest file. In the case of a CSAR VNF package with a TOSCA-Metadata directory, the location, name, and extension of the manifest file shall be specified by means of the "ETSI-Entry-Manifest" keyname in the TOSCA.meta file. In the case of a CSAR VNF package without TOSCA-Metadata directory, the manifest file shall have an extension .mf, the same name as the main TOSCA definitions YAML file and be located at the root of the archive.

The manifest file shall start with the VNF package metadata in the form of a name-value pairs. Each pair shall appear on a different line. The "name" and the "value" shall be separated by a colon and, optionally, one or more blanks. The order of the name-value pairs is not significant.

The name shall be one of those specified in table 4.3.2-1 and the values shall comply with the provisions specified in table 4.3.2-1.

The "required" column in table 4.3.2-1 specifies constraints on the presence of each name in a manifest file. If the cell in the "required" column is set to "Yes", the corresponding name shall be included. If the cell in the "required" column is set to "No", the corresponding name may, but need not to, be included. A name shall not be included more than once.

Table 4.3.2-1: List of valid names and values for VNF package metadata

Name	Value	Required
vnfd_id	A sequence of UTF-8 characters. See note 1.	Yes
vnf_provider_id	A sequence of UTF-8 characters. See note 1.	Yes
vnf_product_name	A sequence of UTF-8 characters. See note 1.	Yes
vnf_release_date_time	A string formatted according to IETF RFC 3339 [3].	Yes
vnf_software_version	A string. See note 1.	Yes
vnf_package_version	A string. See note 2.	Yes
compatible_specification_versions	Indicates which versions of the present document the VNF package complies to, as known at package creation time. See note 3. The value shall be formatted as comma-separated list of strings. Each entry shall have the format <x>.<y>.<z> where <x>, <y> and <z> are decimal numbers representing the version of the present document. Whitespace between list entries shall be trimmed before validation.	Yes See note 4.

Name	Value	Required
vnfm_info	A comma-separated list of strings as defined in the VNFD. Whitespace between list entries shall be trimmed before validation.	Yes
NOTE 1: The value shall be identical to those specified in the VNFD. NOTE 2: The value shall be identical to the descriptor_version attribute specified in the VNFD. NOTE 3: As this list is determined at the time of package creation, it should not be inferred that a package is not compatible with future versions not present in this list. Whether the package will be compatible with such future versions depends on whether these future versions are backward compatible with the listed versions. NOTE 4: A package conformant to versions prior to 2.7.1 does not include this name. Therefore, if this field is missing, it shall be assumed that the package conforms to some previous version of the present document, i.e. a version prior to 2.7.1 and the package shall be considered valid.		

An example of valid manifest file metadata entries follows.

EXAMPLE 1:

```
metadata:
vnfd_id: 2116fd24-83f2-416b-bf3c-ca1964793aca
vnf_product_name: vMRF
vnf_product_name: Virtualized PowerMRF by MyCompany Inc.
vnf_provider_id: MyCompany
vnf_software_version: 1.0.0
vnf_package_version: 1.0
vnf_release_date_time: 2017-01-01T10:00:00+03:00
vnfm_info: etsivnfm:v2.3.1,0:myGreatVnfm-1
compatible_specification_versions: 2.7.1,3.1.1
```

END OF EXAMPLE 1.

The manifest file shall include a list of all files contained in or referenced from the VNF package with their location, expressed using a Source: location/name key-value pair. The manifest file itself may be included in the list.

Below is an example of valid manifest file entries for files contained in or referenced from the VNF package when authenticity and integrity of the VNF package is implemented according to option 1 as specified in clause 5.1.

EXAMPLE 2:

```
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
```

END OF EXAMPLE 2.

If the VNF package is built according to option 1 (clause 5.1), the manifest file shall contain digests of all individual files contained in or referenced from the package.

A consumer of the VNF package verifies the digests in the manifest file by computing the actual digests and comparing them with the digests listed in the manifest file.

The manifest file in option 1 is the key for decision regarding a VNF package integrity and validity in terms of its contained artifacts. The specification of the manifest file and specific algorithms used in digest creation and validation is described in the security related clause.

The details of specifying the local or externally located files and their security protection are described in clause 5.

4.3.3 VNF package change history file

A CSAR VNF package shall have a humanly readable text file describing any change in the constituency of the VNF package. All the changes in the VNF package shall be versioned, tracked and inventoried in the change history file.

In the case of a CSAR VNF package with a TOSCA-Metadata directory, the location, name, and extension of the change history file shall be specified by means of the "ETSI-Entry-Change-Log" keyname in the TOSCA.meta file. In the case of a CSAR VNF package without TOSCA-Metadata directory, the change history file shall be named "ChangeLog.txt" and located at the root of the archive.

4.3.4 VNF package testing files

To enable VNF package validation, a VNF Provider should include in a VNF package files containing necessary information (e.g. test description) in order to perform VNF testing. The contents of VNF testing information are outside the scope of the present document.

In the case of a CSAR VNF package with a TOSCA-Metadata directory, the location and name of a directory containing VNF testing information shall be specified by means of the "ETSI-Entry-Tests" keyname in the TOSCA.meta file. In the case of CSAR VNF package without TOSCA-Metadata directory, the VNF testing information shall be located in a directory named "Tests" located at the root of the archive.

4.3.5 VNF package licensing information

As required in ETSI GS NFV-IFA 011 [1] the VNF package shall contain license information for the released VNF. The license information shall include a single license term for the whole VNF. In addition, the license information may also include license terms for each of the VNF package artifacts if different from the one of the released VNF.

In the case of a CSAR VNF package with a TOSCA-Metadata directory, the location and name of a directory containing VNF licensing information shall be specified by means of the "ETSI-Entry-Licenses" keyname in the TOSCA.meta file. In the case of CSAR VNF package without TOSCA-Metadata directory structure, the VNF licensing information shall be located in a directory named "Licenses" located at the root of the archive.

4.3.6 Certificate file

The CSAR VNF package shall contain a certificate file if the certificate is not included in the signature container (see note) within the manifest file. In this case or if a single certificate is provided for the signature of multiple artifacts (see clause 5.4), the certificate file shall support one of the two following options:

1) In the case of a CSAR VNF package with a TOSCA-Metadata directory, the location, name, and extension of the certificate file shall be specified by means of the "ETSI-Entry-Certificate" keyname in the TOSCA.meta file.

2) In the case of a CSAR VNF package without a TOSCA-Metadata directory, the certificate file shall have an extension .cert and the same name as the main TOSCA definitions YAML file and be located at the root of the archive.

NOTE: Signature container refers to a structure in a standard format (e.g. CMS) which contains signature and additional data needed to process the signature (e.g. certificates, algorithms, etc.).

If the complete CSAR file is signed by the VNF provider (see option 2 in clause 5.1), the certificate file shall be contained in a zip file together with the CSAR file and the signature file if the certificate is not included in the signature file. The certificate file shall have an extension .cert and the same name as the CSAR file.

4.3.7 Non-MANO artifact sets in a VNF package

As required in ETSI GS NFV-IFA 011 [1] the VNF package shall allow to store and identify non-MANO artifact sets in the VNF package file.

Every non-MANO artifact set shall be identified by a non-MANO artifact set identifier which shall be registered in the registry (specified in annex B). A non-MANO artifact set identifier shall be a string that consists of sub-strings which shall not contain characters other than the following: digits (0-9), lowercase ASCII characters (a-z), and the special characters underscore "_" and dash "-". Sub-strings shall be separated by the dot "." character.

All files belonging to the same non-MANO artifact set shall share a common path prefix other than the root of the package.

Non-MANO artifact sets shall be declared at the end of the manifest file. If the package contains at least one non-MANO artifact set, an entry named "non_mano_artifact_sets:" shall be present in the package on its own line after the "metadata" section that is defined in clause 4.3.2. The section defined by the "non_mano_artifact_sets" keyname shall have the following structure:

- Every non-MANO artifact set shall be declared on its own line, by a key name that is equal to the non-MANO artifact set identifier.
- Below the key name, all artifacts that belong to the non-MANO artifact set shall be listed, each on its own line, starting with key name "Source", followed by a colon (":") and, optionally, one or more blanks, and further followed by a file name with path for a file in the CSAR archive that is not contained in the root of this archive.

If the Manifest file provides the integrity assurance of the VNF package (option 1 in clause 5.1), these artifacts shall also appear in the list of blocks of name-value pairs specified in clause 5.3.

An example of the section that declares the non-MANO artifact sets in the package is provided below.

EXAMPLE:

```
non_mano_artifact_sets:
   foo_bar:
      Source: foobar/foo/foo.yaml
      Source: foobar/foo/foo.script
      Source: foobar/bar/descriptor.xml
   prv.happy-nfv.cool:
      Source: happy/cool/123.html
      Source: happy/cool/cool.json
      Source: happy/cool/hot/hot_or_cool.json
```

END OF EXAMPLE.

5 Adding security to TOSCA CSAR

5.1 VNF package authenticity and integrity

As specified in ETSI GS NFV-IFA 011 [1] a VNF package shall support a method for authenticity and integrity assurance.

In order to provide the public key based authenticity and integrity for the whole VNF package one of the two following options shall be followed:

Option 1: The VNF package shall contain a Digest (a.k.a. hash) for each of the components of the VNF package. The table of hashes is included in the manifest file, which is signed with the VNF provider private key. In addition, the VNF provider shall include a signing certificate that includes the VNF provider public key, following a pre-defined naming convention and located either at the root of the archive or in a predefined location (e.g. directory).

The certificate may also be included in the signature container, if the signature format allows that. For example, the CMS format allows to include the certificate in the same container as the signature.

Option 2: The complete CSAR file shall be digitally signed with the VNF provider private key. The VNF provider delivers one zip file consisting of the CSAR file, a signature file and a certificate file that includes the VNF provider public key. Only one signature of the CSAR file shall be present. The certificate may also be included in the signature container, if the signature format allows that.

The manifest shall be signed in both option 1 and option 2. Only one signature of the manifest shall be present.

In option 2, the VNF package delivered would therefore be according to figure 5.1-1.

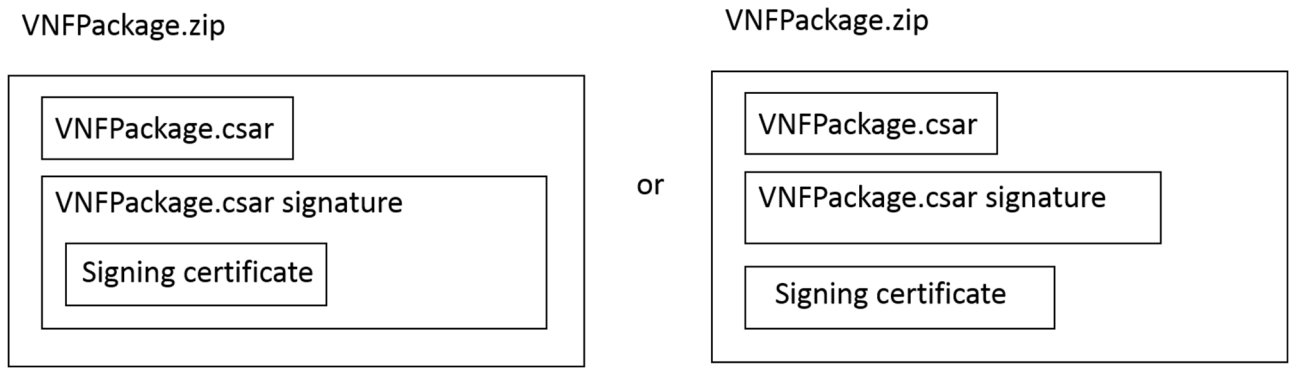

Figure 5.1-1: Composition of the VNF Package zip file in option 2

This solution, either option 1 or option 2, relies on the existence in the NFVO of a root certificate of a trusted CA that shall have been delivered via a trusted channel that preserves its integrity (separate from the VNF package) to the NFVO and be pre-installed in the NFVO before the on-boarding of the VNF package.

NOTE: The present document makes no assumption on who this trusted CA is. Furthermore, it does not exclude that the root certificate be issued by the VNF vendor or by the NFVI provider.

5.2 VNF package manifest and certificate files

In option 1 (see clause 5.1) the manifest file provides the VNF package integrity and authenticity assurance. In this option the manifest contains the digests (hashes) for each individual file locally stored within the VNF package or referenced from it. Each file related entry of the manifest file includes the path or URI of the individual file, the hash algorithm and the generated digest. A consumer of the VNF package shall verify the digests in the manifest file by computing the actual digests and comparing them with the digests listed in the manifest file. Furthermore, the consumer of the VNF Package shall verify that all files in the VNF Package have their corresponding entry and digest in the manifest and that all files listed in the manifest exist in the VNF Package.

In option1 the VNF package authenticity is ensured by signing the manifest file with the VNF provider private key. The digital signature is stored in the manifest file itself (see clause 5.3). The VNF provider shall include an X.509 certificate [8] in the VNF Package. The certificate shall be either placed in a certificate file with extension .cert or, if the chosen signature format allows it, the certificate may be included in the signature container itself. The certificate provides the VNF provider public key.

In option 2 (see clause 5.1), the VNF package authenticity and integrity is ensured by signing the CSAR file with the VNF provider private key (option 2 in clause 5.1) and, if the VNF package refers to external artifacts, with the signature of the external artifacts. The digital signature is stored in a separate file. The VNF provider shall also include an X.509 certificate. The certificate may be included in the signature itself if the signature format allows it or in a separate file. The signature and certificate files shall be siblings of the CSAR file, i.e. placed in the same folder in the parent archive. The signature file shall have an extension .cms and the same name as the CSAR file. Naming conventions for the certificate file are specified in clause 4.3.6. A consumer of the VNF Package built according to option 2 shall verify the signature of the complete package. In addition, a consumer of the VNF package shall verify the signatures of the external artifacts, that are listed in the manifest file, if any.

In this alternative (option 2 in clause 5.1) it is not required to include digests (hashes) per each individual file or artefact in the manifest file. A consumer of the VNF package can verify the signature of the complete CSAR package with the VNF provider public key.

Table 5.2-1 summarizes the characteristics of the two possible options for integrity assurance.

Table 5.2-1: Options for VNF Package integrity assurance: summary of characteristics

Options	Digest per artifact	Signature per artefact	Support external artifacts	Signature as part of the manifest file	External Signature file for the whole CSAR	Certificate may be part of the signature	Certificate may be in a separate file
Option 1	Yes	Yes (mandatory)	Yes	Yes	No	Yes	Yes
Option 2	No	Yes (mandatory)	Yes	Yes	Yes	Yes	Yes

An X.509 certificate included in the VNF package, including those related to the signatures of individual artifacts, may contain one single signing certificate or a complete certificate chain. If a certificate chain is included the signing certificate shall be the first in the list. The rest of the certificates may appear in any order. However, it is recommended that within a chain the certificates are ordered, i.e. each certificate signs the one immediately preceding it in the certificate file. The root certificate that may be present in this X.509 certificate chain shall not be used for validation purposes. Only trusted root certificate pre-installed in NFVO shall be used for validation (see clause 5.1).

Each certificate shall be encoded in a textual representation according to IETF RFC 7468 [6].

5.3 Conventions in the manifest file

When the Manifest file provides the integrity assurance of the VNF package (option 1 in clause 5.1) it shall contain a list of blocks of name-value pairs, where each block is related to one file in the VNF package, where name and value are separated by a colon and, optionally, one or more blanks. Each block shall contain the following name-value pair attributes:

- Source: identifier of the file used as input to the hash generation algorithm. The source can be either:
  - A file name for a file that is contained in the root of the CSAR archive.
  - A file name with path for a file in the CSAR archive that is not contained in the root of this archive.
  - A URI that identifies or references an externally accessible artifact.

NOTE: The URI can merely serve the purpose of identifying the artifact, and the actual access can occur through another URI that is provisioned using the VNF package onboarding procedure specified in ETSI GS NFV-SOL 005 [i.10].

- Algorithm: name of a well-known algorithm used to generate the hash. Including the hash algorithm is optional if it is indicated in the attribute containing the hash.
- Hash: text string corresponding to the hexadecimal representation of the hash.

The value for the Algorithm name-value pair shall be among those registered by IANA for hash function textual names [4]. VNF packages that comply with the present document shall either use "sha-256" or "sha-512".

The signature shall be included at the end of the file. The signature and all necessary data to interpret it (algorithm used to generate the hash and encryption method) shall be included in a structure in a standard format following digital signatures best practices and encoded in a textual representation according to IETF RFC 7468 [6]. The format shall be CMS [5] or PKCS#7 [10]. No character shall be included between the original contents of the manifest file and the structure containing the signature as that would make the validation fail.

Example of valid manifest file entries including manifest signature in CMS format.

EXAMPLE:

```
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-----
MIICpAYJKoZIhvcNAQcCoIIClTCCApECAQMxDTALBglghkgBZQMEAgMwCwYJKoZI
hvcNAQcBMYICbjCCAmoCAQOAFB/IO+R8q7wBvBhoEGC4BeJ1Nw3mMAsGCWCGSAFl
AwQCAzA+BgkqhkiG9w0BAQowMaANMAsGCWCGSAFlAwQCA6EaMBgGCSqGSIb3DQEB
CDALBglghkgBZQMEAgOiBAICAb4EggIABBvNgTx86OcZJcqBz3fXO5DrVkzE3fif
YvRmpfpARs9S/xOxD1OIvxBENTC9i8P4CJHVotiMe3U9Ccioa7xBfhkyb4Zxm7KW
DDcHf19p11Xumv7psbPq+g9WT6ji0aveWFrZItnZ4OFExJhDuQFDXPLP6pWi+8oi
wrVFekgjb5pCXTfshVkEEVNtqFCkn7ZVXdLaHwT3fZRWeT9tJHnvoHyPMrieDA3u
O3XBBMwPj+4gUXWECVxPZd3LW9BA7Ko3Gd7BxL3CXqir5MYLJKt+epdyDm5VpSXi
f/dwDR3L0w+aVJm5bD7DS+H2G0n27ZnFsklVIU4IPjsUvlqbt0KqOkrXiZWjxy+G
cZdbcrlZ9tzZCt5iAWZGNhOakejo0TCZKEJYq4RFnZPKhkstGtcMlVfcy60Os+xI
9o8M4Fj8UDtZFArXIYo7OPW8WDEBrYKdQAmekj3jFFpF+bDLGE9XzjkxJI3DTq/V
caSdgWnxkeZXhIjL0rnnxErAdmNZtg2i+KUUMMxQ9WtTMG77pda0ZHcms8DY6w0T
V6EUynPFuanTpuTT5KRskGQnwtf3d7YzR9MOpkohXz5dPesltDfxALFm7Y7BkUd9
XLpZTOwuMD2dNZy4zmVqz/lY7KXXQanjjiL3dQ2pwx6Od0gMMUHuF5PumuSV74Ku
ntLMN6oDDVQ=
-----END CMS-----
```

END OF EXAMPLE.

5.4 Signature of individual artifacts

The VNF provider shall digitally sign all artifacts individually. Only one signature per artifact shall be present.

If the artifact is included in the package a signature file in CMS [5] or PKCS#7 [10] format shall accompany the artifact.

NOTE 1: A CMS signature generated without signed attributes allows to extract it in raw format, which may be required by some VIM implementations.

Except for the manifest file, the signature file shall be a sibling of the signed artifact, i.e. placed in the same folder in the archive, which could also be the root of the archive. The signature file naming scheme shall be one of the schemes defined below. Within one VNF package, not more than one of the two naming schemes shall be used. The naming schemes are:

- Maintain the original file name and extension and append '.sig' followed by a file type specific one (e.g. '.cms', '.p7b', '.p7c').
- Maintain the original file name but omit the original extension and append '.sig' followed by a file type specific one (e.g. '.cms', '.p7b', '.p7c').

NOTE 2: Maintaining the original extension allows to sign multiples files in the same directory that only differ in their extensions.

If the artifact and signature is included in the package an X.509 certificate shall also be included in the VNF package as per one of the two following alternatives:

1) One certificate per signed artifact, according to one of the following options:

   1a) A certificate file is included as a sibling of the signed artifact file, i.e. placed in the same folder as the signed artifact. The certificate file naming scheme shall be one of the schemes defined below. Within one VNF package, not more than one of the two naming schemes shall be used. The naming schemes are:

   - Maintain the original file name and extension and append '.cert'.
   - Maintain the original name but omit the original extension and append '.cert'.

1b) The certificate is included in the signature file, provided that the signature format allows for it.

Alternative 1 allows to have different certificates per different artifacts, which may be needed e.g. if artifacts contained in the package are signed by 3rd party providers.

2) One single certificate for all signed artifacts in which case the certificate file shall follow the rules specified in clause 4.3.6. If some, but not all, artifacts have an individual signing certificate, the certificate described in this alternative is used only for those artifacts that do not have an individual signing certificate.

Signature and certificate files of the artifacts included in the package shall be listed in their own blocks in the manifest, like any other file.

For external artifacts, delivered using either option 1 or option 2 as defined in clause 5.1, referred to but not included in the package, the signature file in CMS [5] or PKCS#7 [10] format shall be included in the package. The VNF provider shall provide, in addition to those specified in clause 5.3, a name-value pair in the block in the manifest that contains the artifact URI, where name is 'Signature' and value shall be set to the file name with path in the CSAR archive where the signature is contained. The file should have a double extension: '.sig' followed by a file type specific one (e.g. '.cms', '.p7b', '.p7c'). In addition, the signing X.509 certificate [8] shall be provided as per one of the following alternatives:

1) One certificate per signed artifact: included in the signature file, provided the signature format allows for it, or in a certificate file included in the package. The VNF provider shall provide, in addition to those specified in clause 5.3, a name-value pair in the block in the manifest that contains the artifact URI, where name is 'Certificate' and value shall be set to the file name with path in the CSAR archive where the certificate is contained. The extension of the file containing the signing certificate should be '.cert'.

2) One single certificate for all signed artifacts: in one certificate file in the VNF package with extension .cert and the same name as the main TOSCA definitions YAML file and located at the root of the archive (archive without TOSCA-Metadata directory) or in the location specified by the TOSCA.meta file (archive with a TOSCA-Metadata directory). In the latter case, the corresponding entry shall be named "ETSI-Entry-Certificate". If some, but not all, artifacts have an individual signing certificate, the certificate described in this alternative is used only for those artifacts that do not have an individual signing certificate.

Signature and certificate files of external artifacts shall, in addition, be listed in their own blocks in the manifest, like any other file.

These two name-value pairs applicable for blocks in the manifest related to external artifacts, with names 'Signature' and 'Certificate', need not be included for artifacts included in the VNF package because signatures and certificates files for internal artifacts are identified by the naming and placing conventions, i.e. as siblings of the artifact.

Example of a block in the manifest containing entries for an external artifact in option 1.

EXAMPLE 1:

```
Source: https://www.vendor_org.com/MRF/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 36f945953929812aca2701b114b068c71bd8c95ceb3609711428c26325649165
Signature: /somedirectory/somefilename1.sig.cms
Certificate: /somedirectory/somefilename1.cert
```

END OF EXAMPLE 1.

Example of a block in the manifest containing entries for an external artifact in option 2.

EXAMPLE 2:

```
Source: https://www.vendor_org.com/MRF/v4.1/scripts/scale/scale.sh
Signature: /somedirectory/somefilename1.sig.cms
Certificate: /somedirectory/somefilename1.cert
```

END OF EXAMPLE 2.

5.5 Support for security sensitive artifacts

If an artifact is security sensitive, the whole artifact may be encrypted by the VNF provider with an artifact specific key. In case of asymmetric encryption this key is a public key provided by the party who is responsible to on-board and validate the VNF package or to use the artifact, and the VNF provider uses it to encrypt the security sensitive artifact. The consumer of this artifact then decrypts the artifact with its own private key.

In case of symmetric encryption, the public key provided by the party responsible to on-board and validate the VNF package or to use the artifact is used to encrypt a key generated by the VNF provider. The artifact is encrypted with this latter key, which is to be shared with the consumer of the artifact and shall be included in encrypted form in the VNF package. The 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 shall be delivered in CMS format [5]. The CMS file shall provide all necessary information to decrypt the encrypted artifact: algorithm used for the artifact encryption, and in case of symmetric encryption, the encrypted key used for artifact encryption and algorithm used to encrypt the key.

The file name shall have a double extension 'enc.cms'.

The hash of the artifact that is included in the manifest, and the signature that accompanies the artifact, shall be calculated based on the whole CMS file that includes the encrypted artifact.

6 PNFD archive

6.1 General

The PNFD archive format and structure shall follow the same specification as the VNF package (clauses 4 and 5 of the present document) with the differences specified in the following clauses.

In particular the location and naming convention of certificate files, signatures, change history files, non-MANO artifact sets and Entry-definitions file in a PNFD archive follow the same specification as for the VNF package.

6.2 Actors and roles

The roles assumed in clauses 4 and 5 of the present document for a VNF provider in relation to a VNF package are taken by a PNFD archive provider for a PNFD archive.

The present document does not make any assumption on which organization or entity takes the role of a PNFD archive provider.

6.3 PNFD archive file contents

6.3.1 General

A PNFD archive shall contain a main TOSCA definitions YAML file representing all or part of the PNFD, and additional files. It shall be structured according to one of the CSAR structure options described in clause 4.1.1.

NOTE 1: ETSI GS NFV-SOL 001 [i.4] specifies the structure and format of the PNFD based on TOSCA specifications.

NOTE 2: ETSI GS NFV-SOL 006 [i.6] specifies the structure and format of the PNFD based on YANG specifications.

6.3.2 PNFD archive manifest file

A PNFD archive manifest file shall follow the same specification as the manifest file for VNF package except for the meta-data section.

The name in each name-value pair in the PNFD archive metadata shall be one of those specified in table 6.3.2-1 and the values shall comply with the provisions specified in table 6.3.2-1.

The "required" column in table 6.3.2-1 specifies constraints on the presence of each name in a manifest file. If the cell in the "required" column is set to "Yes", the corresponding name shall be included. If the cell in the "required" column is set to "No", the corresponding name may, but need not to, be included. A name shall not be included more than once.

Table 6.3.2-1: List of valid names and values for PNFD archive metadata

Name	Value	Required
pnfd_provider	A sequence of UTF-8 characters See note 1.	Yes
pnfd_name	A sequence of UTF-8 characters See note 1.	Yes
pnfd_release_date_time	String formatted according to IETF RFC 3339 [3].	Yes
pnfd_archive_version	A string See note 2.	Yes
compatible_specification_versions	Indicates which versions of the present document the PNFD archive complies to, as known at PNFD archive creation time. Formatted as comma-separated list of strings. Each entry has the format <x>.<y>.<z> where <x>, <y> and <z> are decimal numbers representing the version of the present document. Whitespace between list entries shall be trimmed before validation. See note 3.	Yes
NOTE 1: The value shall be identical to those specified in the PNFD (provider and name). NOTE 2: The value shall be identical to the version attribute specified in the PNFD. NOTE 3: As this list is determined at the time of PNFD archive creation, it should not be inferred that a PNFD archive is not compatible with future versions not present in this list. Whether the archive will be compatible with such future versions depends on whether these future versions are backward compatible with the listed versions.		

An example of valid manifest file metadata entries follows.

EXAMPLE:

```
metadata:
pnfd_name: MRF
pnfd_provider: SunShineCompany
pnfd_archive_version: 1.0
pnfd_release_date_time: 2017-01-01T10:00:00+03:00
compatible_specification_versions: 2.7.1,3.1.1
```

END OF EXAMPLE.

6.3.3 Not applicable clauses

Clauses 4.3.4 and 4.3.5 are not applicable to a PNFD archive.

The present document does not constraint the type of information that may be included in the PNFD archive as non-MANO artifacts. Thus it does not preclude that files of the nature specified in clauses 4.3.4 and 4.3.5 are included as non-MANO artifacts.

Annex A (informative): TOSCA CSAR examples

A.1 CSAR with the TOSCA-Metadata directory

Below is an example of a CSAR directory structure for NFV including the TOSCA-Metadata, Definitions, Files and Scripts directories. The TOSCA-Metadata directory contains the TOSCA.meta file as specified in [2]. The VNFD (MRF.yaml) and other templates files, if any, are included in the Definitions directory. The Files directory contains the change log file, images and other artifact files. The Scripts directory includes the scripts files that may be called from the VNFD. The manifest file (MRF.mf) is located at the root level of the archive.

EXAMPLE:

```
!------TOSCA-Metadata
         !------TOSCA.meta
         !------TOSCA.meta.sig.cms


!------Definitions
         !----- MRF.yaml
         !----- MRF.yaml.sig.cms (signature)
         !----- OtherTemplates (e.g. type definitions)
         |----- OtherTemplates signatures

!------Files
         !----- ChangeLog.txt
         |----- ChangeLog.txt.sig.cms
         !----- global.cert (global certificate for the package)
         !----- image(s)
         !----- image(s) signature(s)
         !----- other artifacts
         !----- other artifacts signatures
         !------Tests
                 !----- file(s)
                 !----- signature(s)
         !------Licenses
                 !----- file(s)
                 !----- signature(s)
!------Scripts
         !----- install.sh
         !----- install.sh.sig.cms
!----- MRF.mf
```

END OF EXAMPLE.

A.2 CSAR without the TOSCA-Metadata directory

Below is the example of CSAR including the VNFD (MRF.yaml), manifest, certificate, testing, licensing and change log files located at the root level of the CSAR. The Artifacts directory includes an image file and two files that may be called from the VNFD.

EXAMPLE:

```
!--------- MRF.yaml
!--------- MRF.yaml.sig.cms

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

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

!---------  ChangeLog.txt
!---------  ChangeLog.txt.sig.cms

!--------- Tests
```

```
          !----- file(s)
          !----- signature(s)

   !--------- Licenses

          !----- file(s)
          !----- signature(s)

   !--------- Artifacts

          !----- image(s)

          !----- install.sh
          !----- install.sh.sig.cms

          !----- start.yang
          !----- start.yang.sig.cms
```

END OF EXAMPLE.

A.3 CSAR with the YANG VNFD without TOSCA.meta directory

Below is an example of CSAR including the VNFD (CompanyVNFD.xml), a main TOSCA definition YAML file with metadata only (`CompanyVNFD.yaml`), manifest, certificate, licensing and change log files located at the root level of the CSAR. The Scripts directory includes one script file that may be called from the VNFD. The Files directory includes a software image referenced from the VNFD and additional files. This example does not preclude having other YAML files at other locations than the root of the CSAR file.

EXAMPLE:

```
    !——— CompanyVNFD.yaml

    !——— CompanyVNFD.xml

    !——— CompanyVNFD.mf

    !——— CompanyVNFD.cert

!---- ChangeLog.txt

!-----Files

    !---- image(s)

    !—— Instance Data Files
    !—— start.xml

!--------- Licenses

!--------- Scripts

    !----- install.sh
```

END OF EXAMPLE.

Annex B (normative): Non-MANO artifact sets registry

B.1 General

Non-MANO artifact set identifiers shall be registered in the ETSI NFV registry of VNF package or PNFD archive non-MANO artifact sets [i.5]. The registry has a private and a public part.

In the private part of the registry, e.g. vendor and product specific non-MANO artifact sets are registered. The private part is open to anybody.

In the public part, non-MANO artifact sets that have been documented by a standards-developing organization or industry forum in a publicly available specification are registered.

The allocation of the non-MANO artifact set identifiers is made on a first-come first-served basis.

B.2 Non-MANO artifact set identifier format

A non-MANO artifact set identifier shall be a string that shall comply with the following rules:

- For private non-MANO artifact sets , the identifier shall be the concatenation of: `"prv." <registrant> "." <specificPart>`.
- For public non-MANO artifact sets , the identifier shall be: `<specificPart>`.
- <registrant> and <specificPart> shall be strings that comply with the provisions defined in clause 4.3.7.
- <registrant> shall be a string that represents the registrant (e.g. the company or organization name) chosen at registration time.
- <specificPart> shall be a string that represents the non-MANO artifact set. For private non-MANO artifact sets, this string is scoped by <registrant>.

NOTE: The registration authority has the final right to accept or reject <registrant> and <specificPart> strings as part of the registry governance.

B.3 Registered information

The primary elements of the registry are:

Registered identifier

- nonManoArtifactSetId: Identifier of the non-MANO artifact set (mandatory). This identifier includes information whether the identifier is in the "public" or "private" part of the registry.

Registrant information

- Registrant Name: Name of the company or organization registering the non-MANO artifact set (mandatory).
- Previous Registrant Name(s): Name or names of the company or organization to whom the registered identifier has belonged previously, e.g. due to buyout, merger, acquisition (optional).

NOTE: It is assumed that the registration authority will manage further information related to the identity of the registrant (e.g. contact information).

Additional information

- Non-MANO Artifact Set Name: Name of the non-MANO artifact set (any string, mandatory).
- Description: General description of the non-MANO artifact set (any string, optional).
- Specification URI: Publicly reachable URI of the specification that defines the non-MANO artifact set. Needs to be long-lived. (Mandatory for the public part, recommended for the private part).
- Registration Date: Date of the registration (mandatory).

B.4 Initial registration

B.4.1 Template

During initial registration, the information in the template defined in clause B.4.2 shall be provided.

B.4.2 Template

1 Non-MANO artifact set information

Artifact Set Name [M]	<Name of the non-MANO artifact set that is being registered>
Description [O]	<General description of the non-MANO artifact set that is being registered>
Specification reference [M/O]	<Publicly reachable reference (such as a long-lived URI) of a specification that defines the non-MANO artifact set.> Mandatory for the public part, recommended for the private part of the registry

2 Registration information

Registrant name [M]	<Name of the legal entity requesting registration>
Registrant address [M]	<Address of the legal entity requesting registration>
Registrant contact [M]	<Name and email address of the contact person or the function of the legal entity requesting registration>
Registration date [M]	<The date when the registration request was sent>

3 Requested non-MANO artifact set identifier

Either a private or a public identifier but not both shall be registered. The syntax requirements for sub-strings of the registered identifier are defined in clause 4.3.7 and are reproduced here for convenience: Each of these fields may contain digits (0-9), lowercase ASCII characters (a-z), and the special characters underscore "_" and dash "-", and shall not contain any other characters.

3.1 Alternative 1: Private non-MANO artifact set identifier

The private part of the registry is open to anybody, and allows to register, e.g. vendor and product specific non-MANO artifact sets.

Constant		Registrant		Specific part
prv	.		.	

3.2 Alternative 2: Public non-MANO artifact set identifier

The public part of the registry allows to register non-MANO artifact sets that have been documented by a standards-developing organization or industry forum in a publicly available specification.

Specific part

B.5 Registration update

Only limited parts of the registration information are allowed to be updated:

- Registrant name (in case of mergers, etc.). In this case, the registrant information will be updated and the previous registrant information will be preserved in a special "previous registrants" section.
- Registrant contact data (in case of change of contact person).
- Specification URI (in case of update of URI).

Annex C (informative): Bibliography

- IANA register for Media Types.

NOTE: Available at https://www.iana.org/assignments/media-types/media-types.txt.

Annex D (informative): Change History

Date	Version	Information about changes
November 2016	0.0.1	Initial version based on: - NFVSOL(16)000150r2_ETSI_GS_NFV_SOL004_ToC - NFVSOL(16)000151R1_ETSI_GS_NFV_SOL004_Scope - NFVSOL(16)000152R1_ETSI_GS_NFV_SOL004_Normative_References
January 2017	0.0.2	Second version based on: - NFVSOL(17)000025r2_ETSI_GS_NFV_SOL004_Normative text for CSAR Structure options - NFVSOL(16)000188R4_ETSI_GS_NFV_SOL004_Adding content to 4.1 TOSCA YAML Cloud Service Archive (CSAR) Overview - NFVSOL(16)000189R1_ETSI_GS_NFV_SOL004_ Adding Informative References
February 2017	0.0.3	Third version based on: - NFVSOL(17)000018r5_ SOL004 6. Adding Security to TOSCA CSAR - NFVSOL(17)000063r4_ SOL004 Optional Support for Security Sensitive Artifacts - NFVSOL(17)000068r3_ SOL004 CSAR Format and Conventions - NFVSOL(17)000069r2_ SOL004 Consistency between TOSCA CSAR file contents and TOSCA VNFD content - NFVSOL(17)000108r3_ SOL004 - Adding Security to VNF package - merge of contributions 71r1 and 75
March 2017	0.0.4	Fourth version based on: - NFVSOL(17)000117r1_ SOL004 Annex A: CSAR example without the TOSCA-Metadata directory - NFVSOL(17)000070r4_ SOL004 Test Files in VNF package - NFVSOL(17)000140r4_ SOL004 Manifest File Signature - NFVSOL(17)000134r2_SOL004 Annex A: CSAR example with the TOSCA-Metadata directory - NFVSOL(17)000164r1_SOL004 Licensing Agreement in VNF package
May 2017	0.0.5	Fifth version based on: - NFVSOL(17)000248r1_SOL004_-_Editorial_changes - NFVSOL(17)000249r1_SOL004_-_Clause_3_-_Fixing_Definitions_and_Abbreviations - NFVSOL(17)000250_SOL004_-_Clause_5_-_Removal_of_the_clause 5 - NFVSOL(17)000253r2_SOL004_VNF_manifest_metadata - NFVSOL(17)000259r1_SOL004_-_Clause_6_3_-_Hash_Algorithms - NFVSOL(17)000272r3_SOL004_symmetric_encryption - NFVSOL(17)000273r1_SOL004_Description_of_signature_field - NFVSOL(17)000274_SOL004_-_Clause_6_4_-_Removal_of_superfluous_text - NFVSOL(17)000275_SOL004_-_Clause_6_3_-_Miscellaneous_improvements - NFVSOL(17)000277r1_SOL004_-_Clause_4_1_-_Miscellaneous_improvements - NFVSOL(17)000278r1_SOL004_-_Clause_4_2_-_4_3__-_Miscellaneous_improvements - NFVSOL(17)000353_SOL004__Removing_The_Note - NFVSOL(17)000279r3_SOL004_-_Clause_4__-_Clause_6_-_Naming_conventions_and_file_ - NFVSOL(17)000357_SOL004_Naming_Conventions_for_Testing_Files - NFVSOL(17)000290r2_SOL004_Clause_4_3_4_VNF_package_testing - NFVSOL(17)000215r3_SOL004_Signature_of_the_CSAR_file - NFVSOL(17)000216r2_SOL004_Signature_of_individual_artifacts - NFVSOL(17)000251r1_SOL004_-_Annex_A_-_Harmonization

Date	Version	Information about changes
May 2017	0.1.0	Sixth version based on: - NFVSOL(17)000286R2SOL004_Clause_4_3_2_others_VNF_Package_user_consumer - NFVSOL(17)000287R3SOL004_Clause_2_1_normative_reference_missing - NFVSOL(17)000369R2 SOL004 5.4 shared key for symmetric encryption - NFVSOL(17)000184r3 - SOL004 Root Certificate
December 2017	2.3.2	A first maintenance version is based on: - NFVSOL(17)000585r3_SOL004 3rd party VNF Package extensions
April 2018	2.4.2	A second maintenance version is based on: - NFVSOL(18)000092r1_ SOL004ed251: Updates to clause 4.3.2
July 2018	2.4.3	A third maintenance version is based on: - NFVSOL(18)000232_SOL004ed251: Add clarifying note to the description of the vnf_package_version in clause 4.3.2 - NFVSOL(18)000278_ SOL004ed251: Remove semantics of vnf_package_version consistent with SOL001 and SOL003 - NFVSOL(18)000307r2_ SOL004 Registration template - NFVSOL(18)000354r3_ SOL004 Additions and improvements - NFVSOL(18)000435_ SOL004_Multi-files VNFD
November 2018	2.5.2	Release 2 maintenance version is based on: - NFVSOL(18)000668 Correcting a wrong file suffix - NFVSOL(18)000669 Correcting a date/time format - NFVSOL(18)000670 Updating references to TOSCA YAML specs - NFVSOL(18)000671r2 SOL004 - Clarifying the using of a list of artifacts in Manifest file
December 2018	2.5.3	Release 2 maintenance version is based on: - NFVSOL(18)000672r8_SOL004_-_Adding_support_for_YANG_VNFD_
March 2019	2.5.4	Release 2 maintenance version is based on: - NFVSOL(19)000091r5_SOL004ed261_TOSCA_meta_file
May 2019	2.6.2	Release 2 maintenance version is based on: - NFVSOL(18)000746r4_SOL004ed271_Expanded_Scope_for_support_of_PNF_Package
August 2019	2.6.3	Release 2 maintenance version is based on: - NFVSOL(19)000425r1_ SOL004Ed271- Clarification on CSAR options - NFVSOL(19)000426r1_ CSAR_without_TOSCA_Metadata_directory_clarification - NFVSOL(19)000427r4_Manifest_file_rewording
October 2019	2.6.4	Release 2 maintenance version is based on: - NFVSOL(19)000490r2_ SOL004ed271_signature_certificate_files - NFVSOL(19)000491r2_ SOL004ed271_manifest_file_format
November 2019	2.6.5	Release 2 maintenance version is based on: - NFVSOL(19)000746r1_ SOL0043ed271_sequence of encryption hashing and signing - NFVSOL(19)000311r9_ SOL004ed271 - Support for mandatory individual artifact signature - NFVSOL(19)000747r3_ SOL004ed271 version - NFVSOL(19)000718r1_ SOL004ed271 VNF package metadata in manifest - NFVSOL(19)000503r5_ SOL004ed271_security_sensitive_artifacts_text_missing
March 2020	3.0.0	Release 3 maintenance version is based on: - NFVSOL(19)000816r3_ SOL004ed331_adding_reference_of_tosca_simple_profile_yaml_1_3 - NFVSOL(20)000092_ SOL004ed331_release_3_mirror_clarification on reference of TOSCA spec
June 2020	3.0.1	Release 3 maintenance version is based on: - NFVSOL(19)000389_ SOL004ed331 yang_definitions extension usage clarification - NFVSOL(19)000366_ SOL004ed331 Rel-3 mirror License and Testing files 5.4 - NFVSOL(19)000457_ SOL004ed331_Entry_Certificate_editorial_correction - NFVSOL(19)000391_ SOL004ed331_ rel3 mirror corrections on PNFD archive - NFVSOL(19)000515_ SOL004ed331_TOSCA-Meta-File-Version compatibility - NFVSOL(20)000581r14_ SOL004ed331_mirror_of_580_zip_file_compression_method - NFVSOL(20)000583r3_ SOL004ed331 fix to file naming issue

Date	Version	Information about changes
March 2021	3.3.2	Release 3 maintenance version is based on: - NFVSOL(20)000665r1_ SOL004 Annex A - Minor fixes - NFVSOL(21)000005r1_ SOL004ed351 Support of raw signatures - NFVSOL(21)000145_ SOL004ed351 VNF Package validation in option 1 - NFVSOL(21)000146r2_ SOL004ed351 Manifest entries for signatures and certificate files - NFVSOL(21)000147r2_ SOL004ed351 URI to external artifacts - NFVSOL(21)000148r1_ SOL004ed351 Manifest signature - NFVSOL(21)000164_ SOL004ed351 Support of external artifacts in Option 2
September 2021	3.5.2	Release 3 maintenance version is based on: - NFVSOL(21)000354_SOL004ed361 Single signature - NFVSOL(21)000402_ SOL004ed361 certificate chain and format
September 2021	4.0.1	Release 4 maintenance version is based on: - NFVSOL(21)000427_SOL004ed421 Fix attribute name in VNF package metadata note
May 2022	4.0.2	Release 4 maintenance version is based on: - NFVSOL(22)000222_SOL004ed431 Rel-4 mirror of 179 Signed manifest example correction

History

Document history		
V4.3.1	July 2022	Publication