ETSI GS NFV-SOL 007 V2.7.1 (2019-12)

Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Network Service Descriptor File Structure 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-SOL007ed271

Keywords
cloud, data, information model, model, NFV, 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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

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 of the Network Service Descriptor (NSD) file archive and the naming conventions for the different files it contains, fulfilling the requirements specified in ETSI GS NFV-IFA 014 [1] for an NSD file structure.

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 014: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification".

[2] TOSCA-Simple-Profile-YAML-v1.2-csprd01: "TOSCA Simple Profile in YAML Version 1.2".

NOTE: Available at http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.2/csprd01/TOSCA-Simple-Profile-YAML-v1.2-csprd01.pdf.

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

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

[5] IANA register for Hash Function Textual Names.

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

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

[7] Void.

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

[9] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646".

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

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] TOSCA-v1.0-os: "TOSCA Version 1.0".

[i.2] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.3] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[i.4] ETSI GS NFV-SOL 006: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG specification".

[i.5] TOSCA-Simple-Profile-YAML-v1.0, OASIS Standard: "TOSCA Simple Profile in YAML Version 1.0".

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.2] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.2] and the following apply:

CA	Certificate Authority
CMS	Cryptographic Message Syntax
CSAR	Cloud Service ARchive
IANA	Internet Assigned Number Association
PKCS	Public-Key Cryptography Standards
TOSCA	Topology and Orchestration Specification for Cloud Applications
URI	Universal Resource Identifier
UTF	Unicode Transformation Format
YAML	YAML Ain't Markup Language
YANG	Yet Another Next Generation

4 NSD file structure

4.1 TOSCA YAML Cloud Service Archive (CSAR)

4.1.1 CSAR structure

A TOSCA YAML CSAR file is an archive file using the ZIP file format whose structure complies with the TOSCA Simple Profile in YAML version 1.2 specification [2]. According to with the TOSCA Simple Profile YAML version 1.2 specification [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 as defined in TOSCA v1.0 Specification [i.1].

- 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* metadata.

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 used as entry for parsing the contents of the overall CSAR archive.

Any TOSCA definitions files besides the one denoted by the Entry-Definitions keyword can be found by processing respective imports statements in the entry definitions file (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 as described in TOSCA v1.0 Specification [i.1] or pointed to by relative path names through artifact definitions in one of the TOSCA definitions files contained in the CSAR file.

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 1.0 [i.5] 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. Otherwise the CSAR-Version keyword denotes the version 1.0 and all files shall be declared explicitly.

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 clause 4.3.2 to 4.3.5 of the present document are included in the NSD file package and when using CSAR with TOSCA-Metadata directory, as described in clause 4.1.2.1.

NOTE: TOSCA v1.0 Specification [i.1] does not preclude *TOSCA.meta* file block_0 to be extended with key value pair.

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 TOSCA-v1.0 specification [i.1] for the *TOSCA.meta* file. The keynames represents the entries for artifacts defined in clauses 4.3.2 to 4.3.5 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-Certificate	No	string	Location of the Certificate file as defined in clause 4.3.5

NOTE: Use of the Entry-Manifest, Entry-Change-Log, Entry-Tests, and Entry-Certificate keynames defined in version 2.5.1 to 2.6.1 of the present document is deprecated. These keynames are only provided for backward compatibility with legacy NSD file archive consumers; NSD file archive 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: Sunshine.yaml
ETSI-Entry-Manifest: Sunshine.mf
ETSI-Entry-Change-Log: Files/ChangeLog.txt
```

END OF EXAMPLE.

4.1.3 CSAR zip without TOSCA-Metadata directory

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.

4.1.4 TOSCA Entry definition file metadata extension

4.1.4.1 Metadata keynames

Table 4.1.4.1-1 specifies an extension of the list of recognized metadata keynames as specified in TOSCA-Simple-Profile-YAML-v1.2 [2] for the main TOSCA Service Template.

Table 4.1.4.1-1: List of metadata keynames extensions

Keyname	Required	Type	Description
yang_definitions	No	string	Reference to a YANG file representing the NSD within an NSD file archive.

4.1.4.2 Additional requirement

If a YANG-based NSD is included in the NSD file archive, 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 NSD within the NSD file archive. No additional contents shall be included in the main TOSCA definitions YAML file.

EXAMPLE:

```
tosca_definitions_version: tosca_simple_yaml_1_2
metadata:
   template_name: MainServiceTemplate
   template_author: Onboarding portal
   template_version: 1.1
   yang_definitions: Definitions/myNSD.xml
```

END OF EXAMPLE.

4.2 NSD file structure and format

The structure and format of an NSD file archive shall conform to the TOSCA Simple Profile in YAML version 1.2 specification of the CSAR format [2].

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

The consumer of an NSD file archive 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 NSD file contents

4.3.1 General

An NSD file archive shall contain the NSD as a main TOSCA definitions YAML file, representing all or part of the NSD, 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.3] specifies the structure and format of the NSD based on TOSCA specifications.

NOTE 2: ETSI GS NFV-SOL 006 [i.4] specifies the structure and format of the NSD based on YANG specifications.

If the option with a TOSCA-Metadata directory is used and the CSAR-Version parameter indicates version 1.0, all files that are contained in the archive shall be referenced from the TOSCA.meta file. If the CSAR-Version parameter indicates version 1.1, the files that are referenced and pointed to by relative path names through artifact definitions in one of the TOSCA definitions files (e.g. the NSD) contained in the CSAR need not be declared in the TOSCA.meta file.

If a YANG-based NSD is included in the NSD file archive only the option without a TOSCA-Metadata directory is applicable.

Examples of NSD file archive options are described in annex A.

4.3.2 NSD file archive manifest file

A CSAR NSD file archive shall contain a manifest file. In the case of a CSAR NSD file archive 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 NSD file archive 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 NSD file archive 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.

Table 4.3.2-1: List of valid names and values for NSD file archive metadata

Name	Value
nsd_designer	A sequence of UTF-8 [9] characters. See note 1.
nsd_invariant_id	A sequence of UTF-8 [9] characters. See note 1.
nsd_name	A sequence of UTF-8 [9] characters. See note 1.
nsd_release_date_time	String formatted according to IETF RFC 3339 [3].
nsd_file_structure_version	A string. See note 2.
compatible_specification_versions	Indicates which versions of the present document the NSD file archive complies to, as known at file archive 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. If this field is missing, it shall be assumed that the file archive conforms to some previous version of the present document, i.e. a version prior to 2.7.1. Whitespace between list entries shall be trimmed before validation.
NOTE 1: The value shall be identical to that specified in the NSD. NOTE 2: The value shall be identical to the version attribute specified in the NSD. NOTE 3: As this list is determined at the time of file archive creation, it should not be inferred that a file archive is not compatible with future versions not present in this list. Whether the file 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 1:

```
metadata:
nsd_designer: Mycompany
nsd_invariant_id: Sunshine
nsd_name: Sunshine
nsd_file_structure_version: 1.0
nsd_release_date_time: 2018-04-08T10:00+08:00
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 NSD file archive 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 NSD file archive, when authenticity and integrity of the NSD file archive is implemented according to option 1 as specified in clause 5.1.

EXAMPLE 2:

```
Source: SunShine.yaml
Algorithm: SHA-256
Hash: ead2ca54bfd94b72fb210edb67049e8229e07760e7d69d771fea24c159cefda8

Source: scripts/install.sh
Algorithm: SHA-256
Hash: 16bb3cd7c2d685e0b6da9b1f3f67a11efba692d84f78c23f65f73a271be7726f
```

```
Source: https://www.designer_org.com/SunShine/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 94fedf02af0c7f8d4974f0249d85575f167b48e3622bc9791a19eb7d5ce0d5de
```

END OF EXAMPLE 2.

If the NSD file archive refers to external files, the manifest file shall contain digests of all individual files contained in or referenced from the file archive.

If the NSD file archive does not refer to external files, the manifest files may contain digests of the individual files contained in the archive. If the manifest file does not include digests, the complete CSAR file shall be digitally signed by the NS designer. A consumer of the NSD file archive 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, or alternatively, the signature of the CSAR file, is the key for decision regarding an NSD file archive 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 NSD file archive change history file

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

In the case of a CSAR NSD file archive 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 NSD file archive without TOSCA-Metadata directory, the change history file shall be named "ChangeLog.txt" and located at the root of the archive.

4.3.4 Testing files in the NSD file archive

To enable NS validation, an NS designer should include in an NSD file archive, files containing necessary information (e.g. test description) in order to perform NS testing. The contents of NS testing information included in the NSD file archive is outside the scope of the present document.

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

4.3.5 Certificate file

If the manifest file is signed by the NS designer (see option 1 in clause 5.1), the CSAR NSD file archive 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 be supported one of the two following options:

1) In the case of a CSAR NSD file archive 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 NSD file archive 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 NS designer (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.

5 Adding security to TOSCA CSAR

5.1 NSD file archive authenticity and integrity

An NSD file archive shall support a method for authenticity and integrity assurance.

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

Option 1: The NSD file archive shall contain a Digest (a.k.a. hash) for each of files it contains. The table of hashes shall be included in the manifest file, which is signed with the NS designer private key. In addition, the NS designer shall include a signing certificate that includes the NS designer public key, following a predefined 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 NS designer private key. The NS designer delivers one zip file consisting of the CSAR file, a signature file and a certificate file that includes the NS designer public key. 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.

In option 2, the NSD file archive delivered would therefore be structured according to figure 5.1-1.

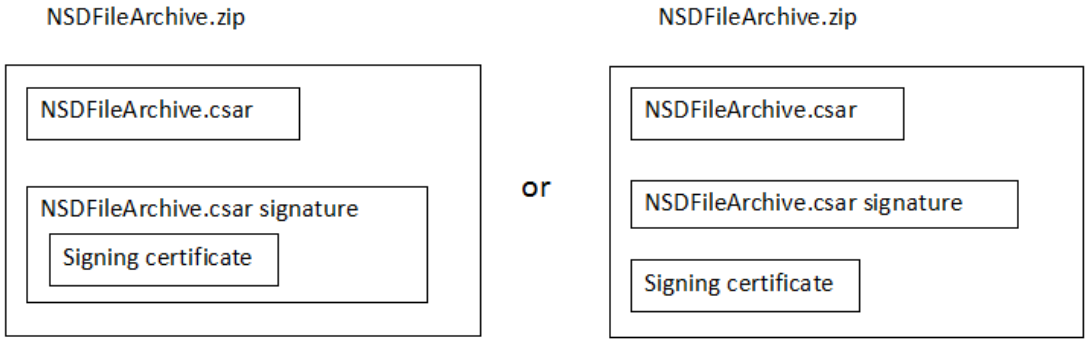

Figure 5.1-1: Composition of the NSD File Archive zip file in option 2

Option 2 is only valid if all artifacts are included in the NSD file archive, i.e. no external artifacts are referenced from the files contained in the NSD file archive.

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 NSD file structure) to the NFVO and be preinstalled in the NFVO before the on-boarding of the NSD file structure.

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 NS designer or by the NFVI provider.

5.2 Manifest and certificate files in the NSD file archive

In option 1 (see clause 5.1) the manifest file provides integrity and authenticity assurance of the NSD file archive. In this option the manifest contains the digests (hashes) for each individual file locally stored within the NSD file archive 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 NSD file archive shall verify the digests in the manifest file by computing the actual digests and comparing them with the digests listed in the manifest file.

In option 1 authenticity of the NSD file archive is ensured by signing the manifest file with the NS designer private key. The digital signature is stored in the manifest file itself (see clause 5.3). The NS designer shall include an X.509 certificate [4] in the NSD file archive. The certificate shall be either placed in a certificate file or, if the chosen signature format allows it, the certificate may be included in the signature container itself. The certificate provides the NS designer public key.

In option 2 (see clause 5.1), authenticity and integrity of the NSD file archive is ensured by signing the CSAR file with the NS designer private key (option 2 in clause 5.1). The digital signature is stored in a separate file. The NS designer 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 signature file. The NS designer creates a zip file consisting 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 4.3.5.

In this alternative (option 2 in clause 5.1) it is not required to include digests (hashes) per each individual file or artifact in the manifest file but it is recommended to include individual signatures of the artifacts (see clause 5.4). A consumer of the NSD file archive can verify the signature of the complete CSAR file with the NS designer public key.

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

Table 5.2-1: Options for NSD file archive integrity assurance: summary of characteristics

Options	Digest per artifact	Signature per artifact	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)	No	Yes	Yes	Yes	Yes

The X.509 certificate may contain one single signing certificate or a complete certificate chain. The root certificate that may be present in this X.509 certificate file shall not be used for validation purposes. Only trusted root certificate pre-installed in NFVO shall be used for validation (see clause 5.1).

5.3 Conventions in the manifest file

When the Manifest file provides the integrity assurance of the NSD file archive (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 NSD file archive, where name and value are separated by a colon and, optionally, one or more blanks. Each block shall contain the following three 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 to an externally accessible artifact.
- 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 [5]. An NSD file archive that complies 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 an archive 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 [8] or PKCS#7 [10].

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

EXAMPLE:

```
Source: SunShine.yaml
Algorithm: SHA-256
Hash: ead2ca54bfd94b72fb210edb67049e8229e07760e7d69d771fea24c159cefda8

Source: scripts/install.sh
Algorithm: SHA-256
Hash: 16bb3cd7c2d685e0b6da9b1f3f67a11efba692d84f78c23f65f73a271be7726f

Source: https://www.designer_org.com/SunShine/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 94fedf02af0c7f8d4974f0249d85575f167b48e3622bc9791a19eb7d5ce0d5de

-----BEGIN CMS-----

MIIFkwYJKoZIhvcNAQcCoIIFhDCCBYACAQExCTAHBgUrDgMCGjCCAacGCSqGSIb3
DQEHAaCCAZgEggGUU291cmNlOiBTdW5TaGluZS55YW1sDQpBbGdvcml0aG06IFNI
QS0yNTYNCkhhc2g6IGVhZDJjYTU0YmZkOTRiNzJmYjIxMGVkYjY3MDQ5ZTgyMjll
MDc3NjBlN2Q2OWQ3NzFmZWEyNGMxNTljZWZkYTgNCg0KU291cmNlOiBzY3JpcHRz
L2luc3RhbGwuc2gNCkFsZ29yaXRobTogU0hBLTI1Ng0KSGFzaDogMTZiYjNjZDdj
MmQ2ODVlMGI2ZGE5YjFmM2Y2N2ExMWVmYmE2OTJkODRmNzhjMjNmNjVmNzNhMjcx
YmU3NzI2Zg0KDQpTb3VyY2U6IGh0dHBzOi8vd3d3LmRlc2lnbmVyX29yZy5jb20v
U3VuU2hpbmUvdjQuMS9zY3JpcHRzL3NjYWxlL3NjYWxlLnNoDQpBbGdvcml0aG06
IFNIQS0yNTYNCkhhc2g6IDk0ZmVkZjAyYWYwYzdmOGQ0OTc0ZjAyNDlkODU1NzVm
MTY3YjQ4ZTM2MjJiYzk3OTFhMTllYjdkNWNlMGQ1ZGWgggIPMIICCzCCAZCgAwIB
AgIJAKGGlg5oMbwkMAoGCCqGSM49BAMCMEMxCzAJBgNVBAYTAmNuMQswCQYDVQQI
DAJqczELMAkGA1UEBwwCbmoxDDAKBgNVBAoMA3p0ZTEMMAoGA1UECwwDaG1mMB4X
DTE4MDMwOTA2NTQyNVoXDTI4MDMwNjA2NTQyNVowQzELMAkGA1UEBhMCY24xCzAJ
BgNVBAgMAmpzMQswCQYDVQQHDAJuajEMMAoGA1UECgwDenRlMQwwCgYDVQQLDANo
bWYwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAASJDqzsB3JidSB1RwqmtR2r12fqnVnp
rvEUQT/wxs+o8JfV7h2ywbXKm3HslE1gGQW+iYkOOPLquhXFp1TdTTrI0SeQ8p1t
Y5RX1Lh7P5su5DfRn5JEMQrhnNhQVOnlDyejUDBOMB0GA1UdDgQWBBSRS//oeOxs
m8wT3QuUHEkRBxRXWjAfBgNVHSMEGDAWgBSRS//oeOxsm8wT3QuUHEkRBxRXWjAM
BgNVHRMEBTADAQH/MAoGCCqGSM49BAMCA2kAMGYCMQC49ePO/17sVE7BMl+B6S78
DYFsSRHkp2RCFgY2Xi9ETdEujY1GgH3yLHz4D2QdRjwCMQDVAL1NhHBeDJR6hSdv
3QIEc8GUYLDjvTxrqEy4OJOgxrVZGV32ZZrH7w+irtH0fdkxggGwMIIBrAIBATBQ
MEMxCzAJBgNVBAYTAmNuMQswCQYDVQQIDAJqczELMAkGA1UEBwwCbmoxDDAKBgNV
BAoMA3p0ZTEMMAoGA1UECwwDaG1mAgkAoYaWDmgxvCQwBwYFKw4DAhqggdgwGAYJ
KoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMTgwOTA3MDMx
NTMzWjAjBgkqhkiG9w0BCQQxFgQU5M/XnNCoeTvXyE22n11tTh9Y30UweQYJKoZI
hvcNAQkPMWwwajALBglghkgBZQMEASowCwYJYIZIAWUDBAEWMAsGCWCGSAFlAwQB
AjAKBggqhkiG9w0DBzAOBggqhkiG9w0DAgICAIAwDQYIKoZIhvcNAwICAUAwBwYF
Kw4DAgcwDQYIKoZIhvcNAwICASgwCQYHKoZIzj0EAQRmMGQCMB7zcw46jub2JhHD
nQ6SGfqvMmBVOE+xDSboJ9WzCeVGzbclO56bNiB0PuwzmYV7ZAIwX/U/2l2lnhAd
/9UqRQKG6z/3WTd/Hm8MpRZSTEan0fPDfrPfU9sunvkV9JQ0Re2Z

-----END CMS-----
```

5.4 Signature of individual artifacts

The NS designer shall digitally sign some or all artifacts individually.

If the artifact is included in the NSD file archive a signature file in CMS [8] or PKCS#7 [10] format shall accompany the signed artifact. Except for the manifest file, the signature file shall have the same name (different extension) as the signed artifact and be a sibling of it, i.e. placed in the same folder in the archive, which could also be the root of the archive. The file shall have a double extension: '.sig' followed by a file type specific one (e.g. '.cms', '.p7b', '.p7c').

If the artifact and signature is included in the NSD file archive an X.509 certificate [4] shall also be included in the NSD file archive as per one of the two following alternatives:

- One certificate per signed artifact:
  Either a certificate file with extension .cert is included as a sibling of the signed artifact file, i.e. placed in the same folder as the signed artifact and having the same name (different extension) or the certificate is included in the signature file, provided that the signature format allows for it. This alternative allows to have different certificates per different artifacts, which may be needed e.g. if artifacts contained in the NSD file archive are signed by 3-rd party designers.

- One single certificate for all signed artifacts, in which case the certificate file shall follow the rules specified in clause 4.3.5. 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.

For external artifacts, delivered using option 1 as defined in clause 5.1, referred to but not included in the NSD file archive, the signature file in CMS [8] or PKCS#7 [10] format shall be included in the NSD file archive. The NS designer 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 [4] shall be provided as per one of the following alternatives:

- 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 NS designer 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'.

- One single certificate for all signed artifacts: in one certificate file in the NSD file archive 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 "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.

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

EXAMPLE:

```
Source: https://www.designer_org.com/SunShine/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 94fedf02af0c7f8d4974f0249d85575f167b48e3622bc9791a19eb7d5ce0d5de
Signature: /somedirectory/somefilename1.sig.cms
Certificate: /somedirectory/somefilename1.cert
```

END OF EXAMPLE.

5.5 Support for security sensitive artifacts

If an artifact is security sensitive, the whole artifact may be encrypted by the NS designer 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 NSD file archive or to use the artifact, and the NS designer 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 NSD file archive or to use the artifact is used to encrypt a key generated by the NS designer. 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 NSD file archive. 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 a CMS format [8]. The CMS file shall provide all necessary information to decrypt 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.

Annex A (informative): TOSCA CSAR Examples

A.1 CSAR with the TOSCA-Metadata directory

Below is an example of a CSAR directory archive including the TOSCA-Metadata, Definitions, Files and Scripts directories. The TOSCA-Metadata directory contains the TOSCA.meta file as specified in [i.1]. The NSD (Sunshine.yaml) and other templates files, if any, are included in the Definitions directory. The Files directory contains the change log file, certificate file and other artifact files. The Scripts directory includes the scripts files that may be called from the NSD. The manifest file (Sunshine.mf) is located at the root level of the archive.

EXAMPLE:

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

!------ Definitions
         !----- SunShine.yaml
         !----- Sunshine.sig.cms (signature)
         !----- Other Templates (e.g., type definitions)
         |----- OtherTemplates signatures

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

END OF EXAMPLE.

A.2 CSAR without the TOSCA-Metadata directory

Below is the example of a CSAR directory structure including the NSD (Sunshine.yaml), manifest, certificate, testing, licensing and change log files located at the root level of the CSAR. The Artifacts directory includes the two scripts files that may be called from the NSD.

EXAMPLE:

```
!--------- Sunshine.yaml
!--------- Sunshine.sig.cms
!--------- Sunshine.mf
!--------- Sunshine.cert
!--------- ChangeLog.txt
!--------- ChangeLog.sig.cms

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

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

!--------- Artifacts
```

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

END OF EXAMPLE.

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

Below is an example of CSAR including the NSD (CompanyNSD.xml), a main TOSCA definition YAML file with metadata only (CompanyNSD.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 NSD. This example does not preclude having other YAML files at other locations than the root of the CSAR file.

EXAMPLE:

```
!——— CompanyNSD.yaml
!——— CompanyNSD.xml
!——— CompanyNSD.mf
!——— CompanyNSD.cert
!---- ChangeLog.txt
!-----Files
        !— Instance Data Files
        !— start.xml
!--------- Licenses
!--------- Scripts
        !----- install.sh
```

END OF EXAMPLE.

Annex B (informative): Bibliography

IANA register for Media Types.

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

Annex C (informative): Change History

Date	Version	Information about changes
April 2018	0.0.1	Initial version based on contributions that were agreed at the NFVSOL#57 meeting and during Email Approval (EA) following the NFVSOL#57. • NFVSOL(18)000105_ETSI_GS_NFV_SOL007_ToC • NFVSOL(18)000106_ETSI_GS_NFV_SOL007_Scope • NFVSOL(18)000107_ETSI_GS_NFV_SOL007_Normative_References
April 2018	0.0.2	Version 0.0.2 based on contributions that were agreed at the NFVSOL#60 meeting and during Email Approval (EA) following the NFVSOL#59 and NFVSOL#60 meetings. • NFVSOL(18)000140_SOL007_Adding_Normative_Reference • NFVSOL(18)000141_SOL007_Normative_text_for_CSAR_Structure_options • NFVSOL(18)000149_SOL007_Adding_4_1_TOSCA_YAML_CSAR • NFVSOL(18)000150_SOL007_Adding_Informative_References • NFVSOL(18)000151_SOL007_Clause_3_-_Fixing_Definitions_and_Abbreviations
August 2018	0.0.3	Version 0.0.3 based on contributions that were agreed at the NFVSOL#74 meeting and during Email Approval (EA) following the NFVSOL#61 and NFVSOL#74 meetings. • NFVSOL(18)000163_SOL007_Adding_NSD_file_contents_General • NFVSOL(18)000468r3_SOL007_NSD_file_structure_manifest_metadata • NFVSOL(18)000476r2_SOL007_Annex_A_CSAR_example_with_the_TOSCA-Metadata_director • NFVSOL(18)000480_SOL007_NSD_file_structure_change_history_file
September 2018	0.1.0	Version 0.1.0 based on contributions that were agreed at the NFVSOL#78 meeting and during Email Approval (EA) following the NFVSOL#75 and NFVSOL#77 meetings. • NFVSOL(18)000484r2_SOL007_NSD_file_structure_testing_files • NFVSOL(18)000509_SOL007_Adding_security_to_TOSCA_CSAR • NFVSOL(18)000512_SOL007_NSD_file_structure_authenticity and integrity_option2 • NFVSOL(18)000513_SOL007_Support_for_security_sensitive_artifacts • NFVSOL(18)000527_SOL007_Conventions_in_the_manifest_file • NFVSOL(18)000531_SOL007_Signature_of_individual_artifacts • NFVSOL(18)000532_SOL007_Certificate_file • NFVSOL(18)000533_SOL007_Clause_5_2_manifest_and_certificate_files
October 2018	0.2.0	Version 0.2.0 based on contributions that were agreed at the NFVSOL#79 meeting and during Email Approval (EA) following the NFVSOL#81 meeting. • NFVSOL(18)000568_SOL007_Annex_A_CSAR_example_without_the_TOSCA-Metadata_director • NFVSOL(18)000597_SOL007_-_Various_Wording_Improvements
January 2019	2.5.2	Version 2.5.2 based on contributions that were agreed during Email Approval (EA) following the NFVSOL#86 meeting. • NFVSOL(18)000756r1_SOL007_Adding_support_for_YANG_NSD • NFVSOL(18)000769_SOL007_Adding_Informative_references_for_YANG_NSD
January 2019	2.5.3	Version 2.5.3 based on contributions that were agreed during Email Approval (EA) following the NFVSOL#87 and the NFVSOL#88 meetings. • NFVSOL(19)000007_SOL007_clarifying_the_using_of_a_list_of_artifacts_in_the_ma • NFVSOL(19)000006_SOL007_correcting_a_date_time_format • NFVSOL(19)000005r1_SOL007_An_improvement_in_clause_4_3_2 • NFVSOL(19)000027_SOL007_An_improvement_for_TOSCA_YAML_version_in_clause_4_1_2
April 2019	2.6.2	Version 2.6.2 based on contributions that were agreed at the NFVSOL#96 meeting and during Email Approval (EA) following the NFVSOL#95 meeting. • NFVSOL(19)000206_SOL007ed271_TOSCA_meta_file • NFVSOL(19)000226_SOL007ed271_Updating_references_to_TOSCA_YAML_specs_in_clause_4_1_2_1

Date	Version	Information about changes
September 2019	2.6.3	Version 2.6.3 based on contributions that were agreed during Email Approval (EA) following the NFVSOL#111 and NFVSOL#112 meetings. • NFVSOL(19)000550_SOL007ed271_Clarification_on_CSAR_options • NFVSOL(19)000555_SOL007ed271_Clarification_on_CSAR_without_TOSCA_Metadata_directory • NFVSOL(19)000558_SOL007ed271_Manifest_file_rewording
November 2019	2.6.4	Version 2.6.4 based on contributions that were agreed at the NFVSOL#117 and NFVSOL#118 meetings and during Email Approval (EA) following the NFVSOL#117 meeting. • NFVSOL(19)000707_SOL007ed271_manifest_file_format • NFVSOL(19)000706r2_SOL007ed271_signature_certificate_files • NFVSOL(19)000745r1_SOL007ed271_security_sensitive_artifacts_text_missing • NFVSOL(19)000771_SOL007ed271_version • NFVSOL(19)000772_SOL007ed271_sequence_of_encryption_hashing_and_signing • NFVSOL(19)000773_SOL007ed271_Support_for_mandatory_individual_artifact_sign Rapporteur action: deleted the authors annex and removed a remaining editor's note.

History

Document history		
V2.5.1	December 2018	Publication
V2.6.1	March 2019	Publication
V2.7.1	December 2019	Publication