ETSI GS NFV-SOL 007 V2.8.1 (2020-08)

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

Reference

RGS/NFV-SOL007ed281

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] OASIS Standard: "TOSCA Simple Profile in YAML Version 1.2".

NOTE: Available at https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.2/TOSCA-Simple-Profile-YAML-v1.2.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".

[11] ISO/IEC 21320-1: "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] 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] Void.

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 [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.

- 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 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 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 v1.2 [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 v1.2 [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 4.3.5 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 Simple Profile YAML v1.2 [2] 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 the present document 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

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

4.1.3.1 General

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 NSD

Table 4.1.3.2-1 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 4.1.3.2-1: 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.

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.

4.1.4 Void

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]. The zip file format shall conform to Document Container Format File [11].

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 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
```