ETSI GS MEC 037 V3.1.1 (2023-03)

GROUP SPECIFICATION

Multi-access Edge Computing (MEC); Application Package Format and Descriptor Specification

Reference

DGS/MEC-0037PKGFORMAT

Keywords

MEC

Contents

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Multi-access Edge Computing (MEC).

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 MEC application package and data models of the MEC application descriptors, fulfilling the requirements defined in ETSI GS MEC 010-2 [1].

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 MEC 010-2: "Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management".

NOTE: The data model in the present document is aligned with the information model specified in V2.2.1 of ETSI GS MEC 010-2 [1].

[2] OASIS Standard TOSCA-Simple-Profile-YAML-v1.3-os: "TOSCA Simple Profile in YAML Version 1.3".

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

[4] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; VNF Package and PNFD Archive specification".

[5] ETSI GS MEC 009: "Multi-access Edge Computing (MEC); General principles, patterns and common aspects of MEC Service APIs".

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

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] ETSI GS MEC 001: "Multi-access Edge Computing (MEC); Terminology".

[i.2] ETSI GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.3] ETSI GS MEC 003: "Multi-access Edge Computing (MEC); Framework and Reference Architecture".

[i.4] ETSI GS MEC 011: "Multi-access Edge Computing (MEC); Edge Platform Application Enablement".

[i.5] ETSI GS MEC 012: "Multi-access Edge Computing (MEC); Radio Network Information API".

[i.6] ETSI GS MEC 013: "Multi-access Edge Computing (MEC); Location API".

[i.7] ETSI GS MEC 014: "Multi-access Edge Computing (MEC); UE Identity API".

[i.8] ETSI GS MEC 015: "Multi-Access Edge Computing (MEC); Traffic Management APIs".

[i.9] ETSI GS MEC 021: "Multi-access Edge Computing (MEC); Application Mobility Service API".

[i.10] ETSI GS MEC 028: "Multi-access Edge Computing (MEC); WLAN Access Information API".

[i.11] ETSI GS MEC 029: "Multi-access Edge Computing (MEC); Fixed Access Information API".

[i.12] ETSI GS MEC 030: "Multi-access Edge Computing (MEC); V2X Information Service API".

[i.13] ETSI GS MEC 033: "Multi-access Edge Computing (MEC); IoT API".

[i.14] ETSI GS MEC 040: "Multi-access Edge Computing (MEC); Federation enablement APIs".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS MEC 001 [i.1], ETSI GR NFV 003 [i.2] and the following apply:

application descriptor: descriptor provided by the application provider which describes the application rules and requirements of a MEC application

application package: bundle of files provided by application provider, on-boarded into MEC system and used by the MEC system for application instantiation, including an application descriptor, a VM image or a URI to a VM image, a manifest file, and other optional files

NOTE: These terms are defined in ETSI GS MEC 010-2 [1].

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS MEC 001 [i.1] and ETSI GR NFV 003 [i.2] apply.

4 Introduction

A MEC application package is a file containing the necessary information of a MEC application, used by the MEC system for application lifecycle management.

The application package, provided by the application provider, typically contains an application descriptor, software images and manifest file as defined in ETSI GS MEC 010-2 [1].

The MEC application package can be on-boarded by the OSS to the MEO. Or in case of MEC in NFV deployment (see ETSI GS MEC 003 [i.3]), the MEC application package can be either on-boarded to the MEAO or to the NFVO.

To unify the MEC package format in both deployment cases, it is defined in such a way that the package can be used by the MEAO directly, and can be on-boarded as a VNF package to the NFVO without any change.

Therefore the MEC application package format is following these principles:

- The MEC application package format is always a valid VNF package.
- Thus the MEC application package contains a valid AppD that is also a valid VNFD.
- Some of the content of the MEC application package and of the AppD will be ignored by the NFV system in the case of MEC in NFV deployment.
- Some of the mandatory VNFD properties and policies are not needed in the MEC system, therefore the AppD will have some fixed default values for those to comply with the VNFD format.

This way, the application developer does not need to know whether the package will be on-boarded to MEO, MEAO or NFVO.

Clauses 5 and 6 of the present document specify the structure and format of the AppD and of the application package based on a TOSCA model (see clause 5.2.1) and a TOSCA YAML CSAR file format (see clause 6.1). Other formats are out of scope of the present document.

5 MEC application descriptor

5.1 MEC application descriptor content

The MEC application descriptor (AppD), including its attributes, is defined in ETSI GS MEC 010-2 [1]. The AppD is provided to the MEC system during onboarding as an artifact of the MEC application package. Since the package format is unified between NFV and MEC, this is described in ETSI GS NFV-SOL 001 [3]. The AppD format is also unified with the VNFD format. MEC attributes that can be mapped to attributes defined by NFV are represented by these NFV attributes, while MEC specific attributes are ignored by the NFV system in case of MEC in NFV deployments.

The following clauses further describe how this is achieved.

5.2 MEC application descriptor based on TOSCA specification

5.2.1 Overview of TOSCA model

TOSCA (Topology and Orchestration Specification for Cloud Applications) is a modelling language for describing the components of a cloud application and their relationships.

There are several versions of TOSCA specifications, and the present document is based on TOSCA-Simple-Profile-yaml v1.3 [2], which describes a YAML rendering for TOSCA.

In TOSCA, the concept "service template" is used to describe the structure and orchestration behaviour of cloud applications. The service template consists of one or multiple "nodes" which describe the components of the application, and "capabilities" and "requirements" which model the dependency between the components. TOSCA also defines artifact, interface, policy types, etc. to describe additional characteristics and requirements of the application.

The service template for each MEC application describes its AppD and shall include a node type which is derived from the MecApp TOSCA node type defined in the present document.

5.2.2 Introduction of TOSCA based AppD model

The format of the AppD is defined a way so it can be used in both MEC standalone and MEC in NFV deployment, and may be on-boarded into NFV-MANO as a VNFD in MEC in NFV deployment. The TOSCA based AppD model specified in the present document is designed to be reusable as a VNFD unchanged, more specifically:

- AppD attributes are mapped to TOSCA VNFD types as defined in ETSI GS NFV-SOL 001 [3] where applicable.
- MEC specific AppD attributes are defined in a way that they will not affect NFV-MANO.
- VNFD mandatory type definitions and properties which are not applicable to AppD are specified to be filled with default values, see clause 5.2.4.1.7 for the default values in tosca.nodes.nfv.VNF and clause 5.2.4.2.2 for default values in other nodes.
- The present document defines an abstract node type (tosca.nodes.mec.MecApp) which is derived from the tosca.nodes.nfv.VNF node type defined in ETSI GS NFV-SOL 001 [3].

Table 5.2.2-1 shows the mapping of the AppD attributes to TOSCA types and property names. AppD attributes that have no corresponding VNFD TOSCA types are MEC specific attributes which are specified in the following clauses.

Table 5.2.2-1: TOSCA types for AppD attributes

AppD attributes	TOSCA types	Property names	Notes
appDId	tosca.nodes.nfv.VNF	descriptor_id	Content same for MEC and NFV, inherit from TOSCA type defined in ETSI GS NFV-SOL 001 [3]
appName	tosca.nodes.nfv.VNF	product_name	
appProvider	tosca.nodes.nfv.VNF	provider	
appSoftVersion	tosca.nodes.nfv.VNF	software_version	
appDVersion	tosca.nodes.nfv.VNF	descriptor_version	
mecVersion	tosca.nodes.mec.MecApp	mec_version	Content as in ETSI GS MEC 010-2 [1]
appInfoName	tosca.nodes.nfv.VNF	product_info_name	Content same for MEC and NFV, inherit from TOSCA type defined in ETSI GS NFV-SOL 001 [3]
appDescription	tosca.nodes.nfv.VNF	product_info_description	
virtualComputeDescriptor	tosca.nodes.nfv.Vdu.Compute	See note 1.	Content same for MEC and NFV, represented by a complete TOSCA type defined in ETSI GS NFV-SOL 001 [3]
swImageDescriptor	tosca.artifacts.nfv.SwImage	See note 1.	
virtualStorageDescriptor	tosca.nodes.nfv.Vdu.VirtualBlockStorage, tosca.nodes.nfv.Vdu.VirtualObjectStorage, tosca.nodes.nfv.Vdu.VirtualFileStorage	See note 1.	
appExtCpd	tosca.nodes.nfv.VduCp	See note 1.	
appServiceRequired	tosca.nodes.mec.MecApp	app_service_required	Content as in ETSI GS MEC 010-2 [1]
appServiceOptional	tosca.nodes.mec.MecApp	app_service_optional	
appServiceProduced	tosca.nodes.mec.MecApp	app_service_produced	
appFeatureRequired	tosca.nodes.mec.MecApp	app_feature_required	
appFeatureOptional	tosca.nodes.mec.MecApp	app_feature_optional	
transportDependencies	tosca.nodes.mec.MecApp	transport_dependencies	
appTrafficRule	tosca.nodes.mec.MecApp	app_traffic_rule	
appDNSRule	tosca.nodes.mec.MecApp	app_dns_rule	
appLatency	tosca.nodes.mec.MecApp	app_latency	
terminateAppInstanceOpConfig	tosca.datatypes.nfv.VnfTerminateOperationConfiguration	See notes 1 and 2.	Content same for MEC and NFV, represented by a complete TOSCA type defined in ETSI GS NFV-SOL 001 [3]
changeAppInstanceStateOpConfig	tosca.datatypes.nfv.VnfOperateOperationConfiguration	See notes 1 and 3.	

AppD attributes	TOSCA types	Property names	Notes
userContextTransferCapability	tosca.nodes.mec.MecApp	user_context_transfer_capability	Content as in ETSI GS MEC 010-2 [1]
appNetworkPolicy	tosca.nodes.mec.MecApp	app_network_policy	

NOTE 1: This attribute of the AppD is represented by a complete TOSCA type and all its properties.
NOTE 2: This attribute of the AppD is represented by the "terminate" property in the tosca.datatypes.nfv.VnfLcmOperationsConfiguration datatype.
NOTE 3: This attribute of the AppD is represented by the "operate" property in the tosca.datatypes.nfv.VnfLcmOperationsConfiguration datatype.

NOTE: The present document does not cover attributes related to MEC federation.

5.2.3 General requirements

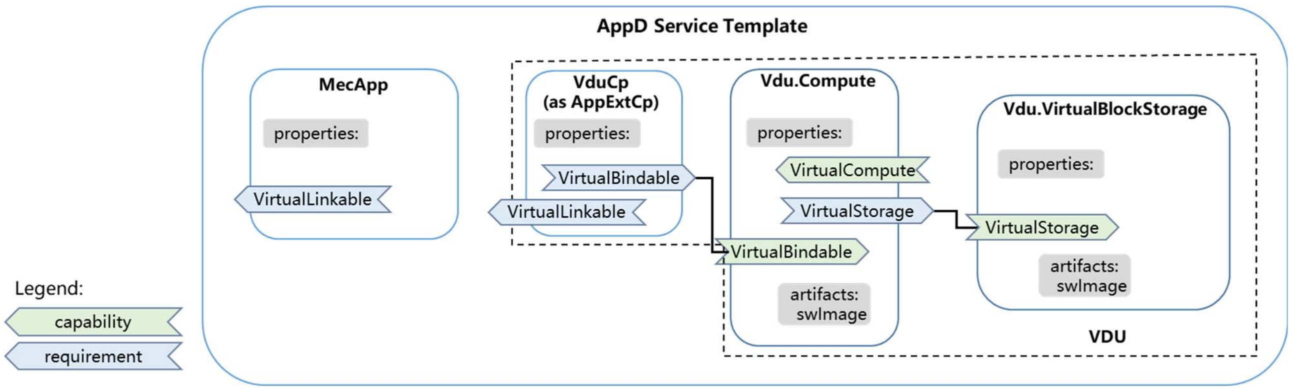
5.2.3.1 Service template design

A TOSCA based AppD can be described in the form of service template.

The structure and requirements of such service template shall conform to the provisions defined in ETSI GS NFV-SOL 001 [3], clause 6.11.3, which describe the single deployment flavour design with one service template, with the following clarifications:

- a node type definition defining a specific node type for the particular MEC app that shall be derived from tosca.nodes.mec.MecApp; either in the form of an inline definition or as import statement referencing a file that contains such definition;
- an import statement referring to tosca.nodes.mec.MecApp. See examples for import statements in clause 5.2.3.2;
- optionally, import statements referencing additional type definitions used by the service template; and
- a topology template describing the topology of the MEC app, with:
  - a node template of the specific node type of the MEC app, see clause 5.2.3.3;
  - further node templates describing the requirements for compute, storage and external network connection, see clause 5.2.3.4;
  - additional definitions and mappings as described in clause 5.2.3.4.

Figure 5.2.3.1-1 provides an overview of TOSCA based AppD service template, including the main node types and relationship between them.



NOTE 1: The external connection points of the MEC app are represented using the VduCp node type according to the mechanism defined in clause A.3.2 of ETSI GS NFV-SOL 001 [3].
NOTE 2: In the present document, a MEC application has no internal virtual links, which means that an external CP always binds to an external virtual link.

Figure 5.2.3.1-1: AppD service template overview

5.2.3.2 Import statement

Import statement is used within a TOSCA service template to reference another file that includes TOSCA type and template definitions. As defined in clause 5.2.3.1, an AppD shall include an import statement that refers to the definition of the specific node type for a particular MEC App or the MEC TOSCA AppD types if the specific node type is defined inline, and additional files which are necessary for processing the AppD. The TOSCA VNFD type definitions are imported via the MecAppDTypes.yaml file. See also annex A.

The TOSCA-Simple-Profile-yaml-v1.3 [2] defines two options to describe import statements: the single-line grammar (clause 3.6.8.2.1) and the multi-line grammar (clause 3.6.8.2.2). The AppD should use the single-line grammar.

The following examples illustrate this:

EXAMPLE 1: AppD service template with single-line grammar import statements which defines the MyMecAppType specific node type inline and which further provides the MecApp specific node type definition:

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of AppD with specific MEC App node type defined inline

imports:
  - https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml
  - other_files.yaml

node_types:
  MyCompany.MyMecAppType.1_0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      ...

topology template:
  ...

  node_templates:
    MyMecApp:
      type: MyCompany.MyMecAppType.1_0:
      ...
```

EXAMPLE 2: AppD service template with single-line grammar import statements which defines the MyMecAppType specific node type in a file for import into other service templates:

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of definitions file

imports:
  - https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml
  - other_files.yaml

node_types:
  MyCompany.MyMecAppType.1_0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      ...
```

EXAMPLE 3: AppD service template for MyMecApp with single-line grammar import statements which imports the type definition from Example 2 as file "MyCompanyMyMecAppType.yaml" and which further provides the MecApp specific node type definition:

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of AppD with specific MEC App node type defined in a separate file

imports:
  - path1/path2/MyCompanyMyMecAppType.yaml

topology template:
  ...

  node_templates:
    MyMecApp:
      type: MyCompany.MyMecAppType.1_0:
```

```
      ...
```

EXAMPLE 4: AppD service template as defined in Example 2 but using the multi-line import grammar:

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of definitions file

imports:
  - file: https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml
  - file: other_files.yaml

node_types:
  MyCompany.MyMecAppType.1_0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      ...
```

5.2.3.3 Requirements on application specific AppD node type

For a given AppD, an application specific node type shall be defined following the below requirements:

a) The node type shall be derived from: tosca.nodes.mec.MecApp (which is derived from tosca.nodes.nfv.VNF).

b) The properties listed in ETSI GS NFV-SOL 001 [3], clause 6.8.1.8, except "flavour_id", shall be included with their values indicated as constraints and as default values.

NOTE: The requirements listed in ETSI GS NFV-SOL 001 [3], clause 6.8.1.8 apply to the application specific node type which will be used for the service template of the MEC application. They are not relevant for the abstract node type tosca.nodes.mec.MecApp, which is not used directly in the service template.

For the AppD, a service template shall be created using the application specific node type.

The following example illustrates the definition of the application specific node type for the example "MyMecApp" MEC application.

EXAMPLE:

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of AppD with tosca.nodes.mec.MecApp node

imports:
  ...

node_types:
  MyCompany.MyMecAppType.1_0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      # defining constraints and default value as required by SOL001
      descriptor_id:
        type: string
        constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: MyMecApp ]
        default: MyMecApp
      software_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      descriptor_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      vnfm_info:
        type: list
        entry_schema:
          type: string
        constraints: [ valid_values: [ [ 'etsivnfm:v2.3.1' ] ] ]
        default: [ 'etsivnfm:v2.3.1' ]
```

5.2.3.4 Requirements on AppD service templates

For the AppD, a service template shall be created using the application specific node type (see clause 5.2.3.3).

The TOSCA service template definitions for the AppD shall include the following parts that are defined in ETSI GS NFV-SOL 001 [3] and of which the details are only shown in examples in the present document.

The following node templates define the resource requirements of the MEC application:

- The AppD service template shall include a node template of tosca.nodes.nfv.Vdu.Compute type which describes the compute requirements of the MEC application, see clause 5.2.4.2 for mandatory default values and more information.
- The AppD service template shall include a node template of tosca.nodes.nfv.VduCp type which describes the requirements on the network connection of the MEC application, see clause 5.2.4.6 for more information.
- The AppD service template may include a node template describing storage requirements of the MEC application, see clauses 5.2.4.3, 5.2.4.4 and 5.2.4.5 for more information.

The following additional definitions and mappings are required:

- The AppD service template shall include substitution mappings that map the network connection to the compute element.
- The AppD service template shall include policy definitions for the instantiation levels, derived from tosca.policies.nfv.InstantiationLevels and from tosca.policies.nfv.VduInstantiationLevels as also explained in clause 5.2.8.

MEC applications need not define application specific values for substitution mappings and instantiation policies. All MEC applications can use the values shown in the examples in the present document. See annex A.

5.2.4 Node type definitions

5.2.4.1 tosca.nodes.mec.MecApp

5.2.4.1.1 Description

The MecApp node type is the generic abstract type from which all MEC application specific node types shall be derived to form, together with other node types, the TOSCA service template(s) representing the AppD information element as defined in ETSI GS MEC 010-2 [1], including the MEC application rules and requirements. Table 5.2.4.1.1-1 specifies the declared names for this node type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.4.1.1-1: Type name, URI, and parent type

Shorthand Name	MecApp
Type Qualified Name	toscamec:MecApp
Type URI	tosca.nodes.mec.MecApp
Parent Type	tosca.nodes.nfv.VNF

5.2.4.1.2 Properties

The properties of the MecApp node type shall comply with the provisions set out in table 5.2.4.1.2-1.

NOTE 1: Inherited properties from tosca.nodes.nfv.VNF are not shown in the below table. They are used as is described in ETSI GS NFV-SOL 001 [3]. For the mapping to attributes of the AppD as described in ETSI GS MEC 010-2 [1], see the table 5.2.2-1. Also the example in clause 5.2.4.1.8 includes inherited properties.

NOTE 2: The requirements listed in ETSI GS NFV-SOL 001 [3], clause 6.8.1.8 are not relevant for this abstract node type. They apply to the derived app specific node type which will be used for the service template of the MEC application. See example in clause 5.2.3.3.

Table 5.2.4.1.2-1: Properties

Name	Required	Type	Constraints	Description
mec_version	yes	string	The value shall be formatted as comma-separated list, e.g.: '1.1.1,1.2.1'. Each entry shall have the format <x>.<y>.<z> where <x>, <y> and <z> are decimal numbers representing the version of ETSI GS MEC 010-2 [1].	Identifies version(s) of MEC system compatible with the MEC application described in this version of the AppD.
app_service_required	no	list of tosca.datatypes.mec.ServiceDependency		Describes the services the MEC application requires to run.
app_service_optional	no	list of tosca.datatypes.mec.ServiceDependency		Describes the services the MEC application may use if available.
app_service_produced	no	list of tosca.datatypes.mec.ServiceDescriptor		Describes services a MEC application is able to produce to the platform or other MEC applications. Only relevant for service-producing apps.
app_feature_required	no	list of tosca.datatypes.mec.FeatureDependency		Describes the features the MEC application requires to run.
app_feature_optional	no	list of tosca.datatypes.mec.FeatureDependency		Describes the features the MEC application may use if available.
transport_dependencies	no	list of tosca.datatypes.mec.TransportDependency		Describes the transport required by the MEC application. The transport is provided by the MEC platform, and is used by the application to deliver services.
app_traffic_rule	no	list of tosca.datatypes.mec.TrafficRuleDescriptor		Describes traffic rules the MEC application requires.
app_dns_rule	no	list of tosca.datatypes.mec.DNSRuleDescriptor		Describes DNS rules the MEC application requires.
app_latency	no	tosca.datatypes.mec.LatencyDescriptor		Describes the maximum latency tolerated by the MEC application.
user_context_transfer_capability	no	tosca.datatypes.mec.UserContextTransferCapability		Describes whether the MEC application supports the user context transfer capability.
app_network_policy	no	tosca.datatypes.mec.AppNetworkPolicy		Describes the application network policy of carrying the application traffic.

5.2.4.1.3 Attributes

None.

5.2.4.1.4 Requirements

See ETSI GS NFV-SOL 001 [3], clause 6.8.1.4 for derived requirements.

5.2.4.1.5 Capabilities

None.

5.2.4.1.6 Definition

The syntax of the MecApp node type shall comply with the following definition:

```
  tosca.nodes.mec.MecApp:
    derived_from: tosca.nodes.nfv.VNF
    description: The generic abstract type from which all specific MEC application node types shall
be derived from, together with other node types, the TOSCA service template(s) representing the AppD
    properties:
      flavour_id: # NFV value common for all MEC applications
        type: string
        constraints: [ equal: simple ]
        default: simple
      flavour_description: # NFV value common for all MEC applications
        type: string
        default: ""
      mec_version:
        type: list
        description: version of the MEC system compatible with the MEC application
        required: true
        entry_schema:
          type: string
          constraints:
            - pattern: ([0-9]?[0-9]\.[0-9]?[0-9]\.[0-9]?[0-9]$)
      app_service_required:
        type: list
        description: services the MEC application requires to run
        required: false
        entry_schema:
          type: tosca.datatypes.mec.ServiceDependency
      app_service_optional:
        type: list
        description: services the MEC application may use if available
        required: false
        entry_schema:
          type: tosca.datatypes.mec.ServiceDependency
      app_service_produced:
        type: list
        description: services a MEC application is able to produce to the platform or other MEC
applications. Only relevant for service-producing apps
        required: false
        entry_schema:
          type: tosca.datatypes.mec.ServiceDescriptor
      app_feature_required:
        type: list
        description: features the MEC application requires to run
        required: false
        entry_schema:
          type: tosca.datatypes.mec.FeatureDependency
      app_feature_optional:
        type: list
        description: features the MEC application may use if available
        required: false
        entry_schema:
          type: tosca.datatypes.mec.FeatureDependency
      transport_dependencies:
        type: list
        description: transport required by the MEC application
        required: false
        entry_schema:
          type: tosca.datatypes.mec.TransportDependency
      app_traffic_rule:
        type: list
        description: traffic rules the MEC application requires
        required: false
        entry_schema:
          type: tosca.datatypes.mec.TrafficRuleDescriptor
      app_dns_rule:
        type: list
        description: DNS rules the MEC application requires
        required: false
        entry_schema:
          type: tosca.datatypes.mec.DNSRuleDescriptor
      app_latency:
        type: tosca.datatypes.mec.LatencyDescriptor
        description: the maximum latency tolerated by the MEC application
        required: false
      user_context_transfer_capability:
        type: tosca.datatypes.mec.UserContextTransferCapability
        description: whether the MEC application supports the user context transfer capability
        required: false
      app_network_policy:
        type: tosca.datatypes.mec.AppNetworkPolicy
        description: the application network policy of carrying the application traffic
        required: false
```

5.2.4.1.7 Additional Requirements

For a given AppD, a node of tosca.nodes.mec.MecApp node type shall be defined following the requirements below:

a) The following required properties shall be included with their default values as indicated in table 5.2.4.1.7-1.

Table 5.2.4.1.7-1: Required properties and default value

Property	Type	Default Value	Note
flavour_id	string	simple	
flavour_description	string	""	empty string

b) Depending on the number of external connection points of the MEC App that need to connect to external virtual links, zero, one or multiple requirements for VirtualLinkable capability shall be defined.

5.2.4.1.8 Example

This example shows the usage of tosca.nodes.mec.MecApp node type. It illustrates the definition of a specific node type for the particular MEC app that is derived from tosca.nodes.mec.MecApp, as well as a definition of the node template for the AppD, using the node type. For a complete definition of the service template see annex A.

NOTE: The virtualComputeDescriptor, swImageDescriptor, virtualStorageDescriptor and appExtCpd parts of the AppD are modelled in separate tosca node types. Examples for these can be found in ETSI GS NFV-SOL 001 [3].

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: example of a specific node type and template for a MEC application

imports:
  - https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml

node_types:
  MyCompany.MyMecApp_1.0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      descriptor_id:
        type: string
        constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: MyMecApp ]
        default: MyMecApp
      software_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      descriptor_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      vnfm_info:
        type: list
        entry_schema:
          type: string
        constraints: [ equal: [ 'etsivnfm:v2.3.1' ] ]
        default: [ 'etsivnfm:v2.3.1' ]
      ... # other application specific properties if any

topology_template:
  ...

  node_templates:
    MyApp:
      type: MyCompany.MyMecApp_1.0
      properties:
        mec_version: [ '2.2.1' ]
        app_service_required:
          - ser_name: RNIS
          version: '1.0'
          requested_permissions:
            - id: query
        app_service_produced:
          - ser_name: AMS
            version: '1.0'
            transports_supported:
              - transport:
                  type: REST_HTTP
                  protocol: HTTP
                  version: '2.0'
                  security:
                     oauth2_grant_type: [ OAUTH2_CLIENT_CREDENTIALS ]
                     oauth2_token_endpoint: 'www.example.com/token'
                serializers:
                  - JSON
        app_feature_required:
          - feature_name: UserApps
            version: '1.0'
        transport_dependencies:
          - transport_serializer:
              transport:
                 type: REST_HTTP
                 protocol: HTTP
                 version: '2.0'
                 security:
                    oauth2_grant_type: [ OAUTH2_CLIENT_CREDENTIALS ]
                    oauth2_token_endpoint: '/token'
              serializers:
                - JSON
            labels:
              - test
        app_traffic_rule:
          - traffic_rule_id: rule1
            filter_type: FLOW
            priority: 1
            traffic_filter:
              - src_address:
                  - 192.168.1.1
                dst_address:
                  - 192.168.1.2
                src_port:
                  - 8080
                dst_port:
                  - 8080
                protocol:
                  - udp
                tag:
                  - 1234
                src_tunnel_address:
                  - 10.10.10.1
                tgt_tunnel_address:
                  - 10.10.10.2
                src_tunnel_port:
                  - 8080
                dst_tunnel_port:
                  - 8080
                qci: 1
                dscp: 0
                tc: 1
            action: DROP
            dst_interface:
              - interface_type: TUNNEL
                tunnel_info:
                  tunnel_type: GTP-U
                  tunnel_dst_address: 10.10.10.2
                  tunnel_src_address: 10.10.10.1
              src_mac_address: 02-00-00-00-00-00
              dst_mac_address: 02-00-00-00-00-01
              dst_ip_address: 192.0.2.1
        app_dns_rule:
          - dns_rule_id: dnsrule1
            domain_name: www.example.com
            ip_address_type: IP_V4
            ip_address: 192.0.2.1
            ttl: 100
        app_latency:
          max_latency: 20000000
        user_context_transfer_capability:
          stateful_application: true
          user_context_transfer_support: true
        app_network_policy:
          cellular_network: true
          wifi_network: false
          fixed_access_network: false
```

5.2.4.2 tosca.nodes.nfv.Vdu.Compute

5.2.4.2.1 Description

The Vdu.Compute node type describes the virtual compute requirements of the MEC application. It refers to the VirtualComputeDescriptor data type in ETSI GS MEC 010-2 [1]. The declared names, properties, requirements, capabilities, etc. for this node type are defined as in clause 6.8.3 in ETSI GS NFV-SOL 001 [3].

5.2.4.2.2 Additional Requirements

When used in an AppD, the node description shall include the following required properties with their default values set as defined in table 5.2.4.2.2-1.

Table 5.2.4.2.2-1: Required properties and default value

Property	Type	Default Value	Note
name	string	""	empty string
description	string	""	empty string
boot_order	boolean	false	a MEC application is always deployed as a single instance
vdu_profile	tosca.datatypes.nfv.VduProfile	min_number_of_instances: 1 max_number_of_instances: 1	

5.2.4.2.3 Example

Example usage of node template description:

```
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  ...

topology template:
  ...

  node_templates:
    Compute:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: ""
        description: ""
        boot_order: false
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 1
      capabilities:
        virtual_compute:
          properties:
            virtual_cpu:
              cpu_architecture: x86
              number_virtual_cpu: 2
            virtual_memory:
              virtual_mem_size: 8192 MiB
              numa_enabled: true
      requirements:
        - virtual_storage: Storage
```

5.2.4.3 tosca.nodes.nfv.Vdu.VirtualBlockStorage

5.2.4.3.1 Description

The VirtualBlockStorage node type describes the requirements related to virtual block storage resources. The declared names, properties, requirements, capabilities, etc. for this node type are defined as in clause 6.8.4 in ETSI GS NFV-SOL 001 [3].

5.2.4.3.2 Example

Example usage of node template description:

```
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  ...

topology template:
  ...

  node_templates:
    BlockStorage:
      type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
      properties:
        virtual_block_storage_data:
          size_of_storage: 100 GB
          rdma_enabled: true
```

5.2.4.4 tosca.nodes.nfv.Vdu.VirtualObjectStorage

5.2.4.4.1 Description

The VirtualObjectStorage node type describes the requirements related to virtual object storage resources. The declared names, properties, requirements, capabilities, etc. for this node type are defined as in clause 6.8.5 in ETSI GS NFV-SOL 001 [3].

5.2.4.4.2 Example

Example usage of node template description:

```
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  ...

topology template:
  ...

  node_templates:
    ObjectStorage:
      type: tosca.nodes.nfv.Vdu.VirtualObjectStorage
      properties:
        virtual_object_storage_data:
        max_size_of_storage: 100 GB
```

5.2.4.5 tosca.nodes.nfv.Vdu.VirtualFileStorage

5.2.4.5.1 Description

The VirtualFileStorage node type describes the requirements related to virtual file storage resources. The declared names, properties, requirements, capabilities, etc. for this node type are defined as in clause 6.8.6 in ETSI GS NFV-SOL 001 [3].

5.2.4.5.2 Example

Example usage of node template description:

```
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  ...

topology template:
  ...

  node_templates:
    FileStorage:
      type: tosca.nodes.nfv.Vdu.VirtualFileStorage
      properties:
        virtual_file_storage_data:
          size_of_storage: 100 GB
          file_system_protocol: NFS
```

5.2.4.6 tosca.nodes.nfv.VduCp

5.2.4.6.1 Description

The VduCp node type describes the requirements related to the external connection points of the MEC application. The declared names, properties, requirements, capabilities, etc. for this node type are defined as in clause 6.8.8 in ETSI GS NFV-SOL 001 [3].

5.2.4.6.2 Example

Example usage of node template description:

```
tosca_definitions_version: tosca_simple_yaml_1_3

imports:
  ...

topology template:
  ...

  node_templates:
    ExtCp_1:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv4 ]
      requirements:
        - virtual_binding: Compute
```

5.2.5 Data type definitions

5.2.5.1 Introduction

This clause defines the data types which are used to describe the properties in the node types defined in clause 5.2.4 and data types used within.

5.2.5.2 tosca.datatypes.mec.ServiceDependency

5.2.5.2.1 Description

The ServiceDependency data type describes the requirements of a service-consuming MEC application to a MEC service, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.2.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.2.1-1: Type name, shorthand and URI

Shorthand Name	ServiceDependency
Type Qualified Name	toscamec:ServiceDependency
Type URI	tosca.datatypes.mec.ServiceDependency

5.2.5.2.2 Properties

The properties of the ServiceDependency data type shall comply with the provisions set out in table 5.2.5.2.2-1.

Table 5.2.5.2.2-1: Properties

Name	Required	Type	Constraints	Description
ser_name	yes	string		The name of the service, see note.
ser_category	no	tosca.datatypes.mec.CategoryRef		A category reference of the service.
version	yes	string		The version of the service.
ser_transport_dependencies	no	list of tosca.datatypes.mec.TransportDependency		Indicates transport and serialization format dependencies of consuming the service.
requested_permissions	no	list of datatypes.mec.Permission		Requested permissions regarding the access of the application to the service, see note.
NOTE: For ETSI MEC defined services, the value of the attribute refers to table 5.2.5.2.2-2.				

The ser_name and version of the service dependencies correspond to the ser_name in the service descriptor (clause 5.2.5.3) for user-defined services. These services are registered to the MEC system as described in ETSI GS MEC 011 [i.4] using these values.

The MEC system defines the following services that can be referenced by the ser_name given in table 5.2.5.2.2-2.

The version of these services is the mec_version as in table 5.2.4.1.2-1.

Table 5.2.5.2.2-2 also provides the permitted values for requested_permissions for the MEC defined services by reference to the respective documents.

Table 5.2.5.2.2-2: MEC defined services

Name	ser_name	Document defining the service	Values for permissions
Radio Network Information Service	RNIS	ETSI GS MEC 012 [i.5]	The permission identifiers of RNIS are documented in ETSI GS MEC 012 [i.5], annex A.
Location Service	LS	ETSI GS MEC 013 [i.6]	ETSI GS MEC 013 [i.6] does not define permissions.
UE Identity Service	UIS	ETSI GS MEC 014 [i.7]	ETSI GS MEC 014 [i.7] does not define permissions.
BandWidth Management Service	BWMS	ETSI GS MEC 015 [i.8]	ETSI GS MEC 015 [i.8] does not define permissions.
Traffic Management Service	TMS	ETSI GS MEC 015 [i.8]	ETSI GS MEC 015 [i.8] does not define permissions.
Application Mobility Service	AMS	ETSI GS MEC 021 [i.9]	The permission identifiers of AMS are documented in ETSI GS MEC 021 [i.9], clause A.2.
WLAN Access Information Service	WAIS	ETSI GS MEC 028 [i.10]	The permission identifiers of WAIS are documented in ETSI GS MEC 028 [i.10], annex B.
Fixed Access Information Service	FAIS	ETSI GS MEC 029 [i.11]	The permission identifiers of FAIS are documented in ETSI GS MEC 029 [i.11], annex A.
V2X Information Service	VIS	ETSI GS MEC 030 [i.12]	The permission identifiers of VIS are documented in ETSI GS MEC 030 [i.12], annex A.
IoT Service	IOTS	ETSI GS MEC 033 [i.13]	ETSI GS MEC 033 [i.13] does not define permissions.
Federation Enablement Service	FES	ETSI GS MEC 040 [i.14]	ETSI GS MEC 040 [i.14] does not define permissions.

5.2.5.2.3 Definition

The syntax of the ServiceDependency data type shall comply with the following definition:

```
tosca.datatypes.mec.ServiceDependency:
  derived_from: tosca.datatypes.Root
  description: The requirements of a service-consuming MEC application to a MEC service
  properties:
    ser_name:
      type: string
      description: name of the service, for example, RNIS, LocationService, etc.
      required: true
    ser_category:
      type: tosca.datatypes.mec.CategoryRef
      description: a category reference of the service
      required: false
    version:
      type: string
      description: version of the service
      required: true
    ser_transport_dependencies:
      type: list
      description: the transport and serialization format dependencies of consuming the service
      required: false
      entry_schema:
        type: tosca.datatypes.mec.TransportDependency
    requested_permissions:
      type: list
      description: requested permissions regarding the access of the application to the service
      required: false
      entry_schema:
        type: tosca.datatypes.mec.Permission
```

5.2.5.2.4 Example

```
<some_tosca_entity>:
  properties:
    app_service_required:
       ser_name: RNIS
       ser_category:
         …
       version: '1.0'
       ser_transport_dependencies:
         - …
       requestedPermissions:
         - …
```

5.2.5.2.5 Additional Requirements

None.

5.2.5.3 tosca.datatypes.mec.ServiceDescriptor

5.2.5.3.1 Description

The ServiceDescriptor data type describes a MEC service produced by a service-providing MEC application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.3.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.3.1-1: Type name, shorthand and URI

Shorthand Name	ServiceDescriptor
Type Qualified Name	toscamec:ServiceDescriptor
Type URI	tosca.datatypes.mec.ServiceDescriptor

5.2.5.3.2 Properties

The properties of the ServiceDescriptor data type shall comply with the provisions set out in table 5.2.5.3.2-1.

Table 5.2.5.3.2-1: Properties

Name	Required	Type	Constraints	Description
ser_name	yes	string		The name of the service, for example, RNIS, LocationService, etc.
ser_category	no	tosca.datatypes.mec.CategoryRef		A category reference of the service.
version	yes	string		The version of the service.
transport_supported	no	list of tosca.datatypes.mec. TransportSerializerDescriptor		Indicates transport and serialization format supported by the service produced by the application.

5.2.5.3.3 Definition

The syntax of the ServiceDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.ServiceDescriptor:
    derived_from: tosca.datatypes.Root
    description: a MEC service produced by a service-providing MEC application
    properties:
      ser_name:
        type: string
        description: name of the service, for example, RNIS, LocationService, etc.
        required: true
      ser_category:
        type: tosca.datatypes.mec.CategoryRef
        description: a category reference of the service
        required: false
      version:
        type: string
        description: version of the service
        required: true
      transport_supported:
        type: list
        description: transport and serialization format supported by the service produced by the
application
        required: false
        entry_schema:
          type: tosca.datatypes.mec.TransportSerializerDescriptor
```

5.2.5.3.4 Example

```
<some_tosca_entity>:
  properties:
    app_service_produced:
      ser_name: RNIS
      ser_category:
        …
      version: '1.0'
      transport_supported:
        - …
```

5.2.5.3.5 Additional Requirements

None.

5.2.5.4 tosca.datatypes.mec.FeatureDependency

5.2.5.4.1 Description

The FeatureDependency data type describes the requirements of a MEC application to a feature of the MEC platform, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.4.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.4.1-1: Type name, shorthand and URI

Shorthand Name	FeatureDependency
Type Qualified Name	toscamec:FeatureDependency
Type URI	tosca.datatypes.mec.FeatureDependency

5.2.5.4.2 Properties

The properties of the FeatureDependency data type shall comply with the provisions set out in table 5.2.5.4.2-1.

Table 5.2.5.4.2-1: Properties

Name	Required	Type	Constraints	Description
feature_name	yes	string		The name of the feature, for example, UserApps, UEIdentity, etc.
version	yes	string		The version of the feature.

5.2.5.4.3 Definition

The syntax of the FeatureDependency data type shall comply with the following definition:

```
  tosca.datatypes.mec.FeatureDependency:
    derived_from: tosca.datatypes.Root
    description: requirements of a MEC application to a feature of the MEC platform
    properties:
      feature_name:
        type: string
        description: name of the feature, for example, UserApps, UEIdentity, etc.
        required: true
      version:
        type: string
        description: version of the feature
        required: true
```

5.2.5.4.4 Example

```
<some_tosca_entity>:
  properties:
    app_feature_required:
      feature_name: UserApps
      version: '1.0'
```

5.2.5.4.5 Additional Requirements

None.

5.2.5.5 tosca.datatypes.mec.TransportDependency

5.2.5.5.1 Description

The TransportDependency data type describes the requirements of a MEC application to the transport bindings (each being a combination of a transport with one or more serializers), as defined in ETSI GS MEC 010-2 [1].
Table 5.2.5.5.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.5.1-1: Type name, shorthand and URI

Shorthand Name	TransportDependency
Type Qualified Name	toscamec:TransportDependency
Type URI	tosca.datatypes.mec.TransportDependency

5.2.5.5.2 Properties

The properties of the TransportDependency data type shall comply with the provisions set out in table 5.2.5.5.2-1.

Table 5.2.5.5.2-1: Properties

Name	Required	Type	Constraints	Description
transport_serializer	yes	tosca.datatypes.mec.TransportSerializerDescriptor		Information about the transport in this transport binding.
labels	yes	list of string		Set of labels that allow to define groups of transport bindings. The mechanism of the grouping is defined in clause 5.2.5.5.5.

5.2.5.5.3 Definition

The syntax of the TransportDependency data type shall comply with the following definition:

```
  tosca.datatypes.mec.TransportDependency:
    derived_from: tosca.datatypes.Root
    description: requirements of a MEC application to the transport bindings
    properties:
      transport_serializer:
        type: tosca.datatypes.mec.TransportSerializerDescriptor
        description: information about the transport in this transport binding
        required: true
      labels:
        type: list
        description: set of labels that allow to define groups of transport bindings
        required: true
        entry_schema:
          type: string
```

5.2.5.5.4 Example

```
<some_tosca_entity>:
  properties:
    transport_dependencies:
      - transport_serializer:
            …
                labels:
          - test
```

5.2.5.5.5 Additional Requirements

The "labels" property can be used to identify groups of transport bindings: in a list of TransportDependency structures, all entries that have a "labels" entry with the same value belong to the same group. Each group indicates an alternative set of transport bindings. At least one group of transport bindings needs to be supported to fulfil the requirements.

EXAMPLE 1: An application requires REST_HTTP transport with JSON.

List of TransportDependency structures:

```
transport_dependencies:
  - transport_serializer:
      transport:
        type: REST_HTTP
        …
      serializers:
        - JSON
    labels:
      - A
```

EXAMPLE 2: An application can run with JSON or PROTOBUF3 over a topic-based message bus.

List of TransportDependency structures:

```
transport_dependencies:
  - transport_serializer:
      transport:
        type: MB_TOPIC_BASED
        …
      serializers:
        - JSON
        - PROTOBUF3
    labels:
      - A
```

EXAMPLE 3: An application requires REST transport with JSON or a topic-based message bus with PROTOBUF3.

List of TransportDependency structures:

```
transport_dependencies:
  - transport_serializer:
      transport:
        type: REST_HTTP
        …
      serializers:
        - JSON
    labels:
      - A
  - transport_serializer:
      transport:
        type: MB_TOPIC_BASED
        …
      serializers:
        - PROTOBUF3
    labels:
      - B
```

EXAMPLE 4: An application requires both REST transport with JSON and a topic-based message bus with PROTOBUF3.

List of TransportDependency structures:

```
transport_dependencies:
  - transport_serializer:
      transport:
        type: REST_HTTP
        …
      serializers:
        - JSON
    labels:
      - A
  - transport_serializer:
      transport:
        type: MB_TOPIC_BASED
        …
      serializers:
        - PROTOBUF3
    labels:
      - A
```

EXAMPLE 5: An application requires REST transport with JSON and at least one transport among a topic-based message bus with PROTOBUF3 or Websockets with PROTOBUF3.

List of TransportDependency structures:

```
transport_dependencies:
  - transport_serializer:
      transport:
        type: REST_HTTP
        …
      serializers:
        - JSON
    labels:
      - A
      - B
  - transport_serializer:
      transport:
        type: MB_TOPIC_BASED
        …
      serializers:
        - PROTOBUF3
    labels:
      - A
  - transport_serializer:
      transport:
        type: WEBSOCKETS
        …
      serializers:
        - PROTOBUF3
    labels:
      - B
```

5.2.5.6 tosca.datatypes.mec.TrafficRuleDescriptor

5.2.5.6.1 Description

The TrafficRuleDescriptor data type describes traffic rules related to a MEC application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.6.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.6.1-1: Type name, shorthand and URI

Shorthand Name	TrafficRuleDescriptor
Type Qualified Name	toscamec:TrafficRuleDescriptor
Type URI	tosca.datatypes.mec.TrafficRuleDescriptor

5.2.5.6.2 Properties

The properties of the TrafficRuleDescriptor data type shall comply with the provisions set out in table 5.2.5.6.2-1.

Table 5.2.5.6.2-1: Properties

Name	Required	Type	Constraints	Description
traffic_rule_id	yes	string		Identifier of the traffic rule.
filter_type	yes	string	valid value: [FLOW, PACKET]	Definition of filter type: per FLOW or PACKET If it is per FLOW, the filter matches UE to core network packets and the reverse packets are handled by the same context.
priority	yes	integer	greater_or_equal: 0	Priority of this traffic rule. If traffic rule conflicts, the one with higher priority take precedence. The value '0' expresses the highest priority.
traffic_filter	yes	list of tosca.datatypes. mec.TrafficFilter		The filter used to identify specific flow/packets that need to be handled by the MEC host.
action	yes	string	valid_values: [DROP, FORWARD_DECAPSULATED, FORWARD_AS_IS, PASSTHROUGH, DUPLICATED_DECAPSULATED, DUPLICATE_AS_IS]	Identifies the action of the MEC host data plane, when a packet matches the trafficFilter.
dst_interface	no	list of tosca.datatypes. mec.InterfaceDescriptor	max_length: 2	Describes the destination interface information, if the action is FORWARD. Some applications (e.g. inline/tap) require two interfaces, where the first is on the UE side and the second is on the core network side.

5.2.5.6.3 Definition

The syntax of the TrafficRuleDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.TrafficRuleDescriptor:
    derived_from: tosca.datatypes.Root
    description: traffic rules related to a MEC application
    properties:
      traffic_rule_id:
        type: string
        description: identifier of the traffic rule.
        required: true
      filter_type:
        type: string
        description: filter type
        required: true
        constraints:
          - valid_values: [ FLOW, PACKET ]
      priority:
        type: integer
        description: priority of this traffic rule.
        required: true
        constraints:
          - greater_or_equal: 0
      traffic_filter:
        type: list
        description: the filter used to identify specific flow/packets that need to be handled by
the MEC host.
        required: true
        entry_schema:
          type: tosca.datatypes.mec.TrafficFilter
      action:
        type: string
        description: the action of the MEC host data plane, when a packet matches the
traffic_filter.
```

```
        required: true
        constraints:
          - valid_values: [ DROP, FORWARD_DECAPSULATED, FORWARD_AS_IS, PASSTHROUGH,
DUPLICATED_DECAPSULATED, DUPLICATE_AS_IS ]
      dst_interface:
        type: list
        description: describes the destination interface information, if the action is FORWARD.
        required: false
        entry_schema:
          type: tosca.datatypes.mec.InterfaceDescriptor
        constraints:
          - max_length: 2
```

5.2.5.6.4 Example

```
<some_tosca_entity>:
  properties:
    app_traffic_rule:
    - traffic_rule_id: TrafficRule1
      filter_type: FLOW
      priority: 1
      traffic_filter:
        - …
      action: FORWARD
      dst_interface:
        …
```

5.2.5.6.5 Additional Requirements

None.

5.2.5.7 tosca.datatypes.mec.DNSRuleDescriptor

5.2.5.7.1 Description

The DNSRuleDescriptor data type describes DNS rules associated with a MEC application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.7.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.7.1-1: Type name, shorthand and URI

Shorthand Name	DNSRuleDescriptor
Type Qualified Name	toscamec:DNSRuleDescriptor
Type URI	tosca.datatypes.mec.DNSRuleDescriptor

5.2.5.7.2 Properties

The properties of the DNSRuleDescriptor data type shall comply with the provisions set out in table 5.2.5.7.2-1.

Table 5.2.5.7.2-1: Properties

Name	Required	Type	Constraints	Description
dns_rule_id	yes	string		Identifies the DNS Rule.
domain_name	yes	string		FQDN of the DNS rule.
ip_address_type	yes	string	valid values: [IP_V6, IP_V4]	Specifies the IP address type.
ip_address	yes	string		IP address given by the DNS rule.
ttl	no	integer	greater_than: 0	Time-to-live value, in seconds.

5.2.5.7.3 Definition

The syntax of the DNSRuleDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.DNSRuleDescriptor:
    derived_from: tosca.datatypes.Root
    description: DNS rules associated with a MEC application
    properties:
      dns_rule_id:
        type: string
        description: identifier of the DNS rule
        required: true
      domain_name:
        type: string
        description: FQDN of the DNS rule
        required: true
      ip_address_type:
        type: string
        description: IP address type
        required: true
        constraints:
          - valid_values: [ IP_V6, IP_V4 ]
      ip_address:
        type: string
        description: IP address given by the DNS rule
        required: true
      ttl:
        type: integer
        description: time-to-live value
        required: true
        constraints:
          - greater_than: 0
```

5.2.5.7.4 Example

```
<some_tosca_entity>:
  properties:
    app_dns_rule:
      - dns_rule_id: dnsRule1
        domain_name: 'www.example.com'
        ip_address_type: IP_V4
        ip_address: 192.0.2.1
        ttl: 1800
```

5.2.5.7.5 Additional Requirements

None.

5.2.5.8 tosca.datatypes.mec.LatencyDescriptor

5.2.5.8.1 Description

The LatencyDescriptor data type describes latency requirements of a MEC application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.8.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.8.1-1: Type name, shorthand and URI

Shorthand Name	LatencyDescriptor
Type Qualified Name	toscamec:LatencyDescriptor
Type URI	tosca.datatypes.mec.LatencyDescriptor

5.2.5.8.2 Properties

The properties of the LatencyDescriptor data type shall comply with the provisions set out in table 5.2.5.8.2-1.

Table 5.2.5.8.2-1: Properties

Name	Required	Type	Constraints	Description
max_latency	yes	integer	greater_than: 0	The value of the maximum latency in nano seconds tolerated by the MEC application. The latency is considered to be the one way end-to-end latency between the client application (e.g. in a device) and the service (i.e. the MEC Application instance).

5.2.5.8.3 Definition

The syntax of the LatencyDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.LatencyDescriptor:
    derived_from: tosca.datatypes.Root
    description: latency requirements of a MEC application
    properties:
      max_latency:
        type: integer
        description: maximum latency (in nano seconds) tolerated by the MEC application.
        required: true
        constraints:
          - greater_than: 0
```

5.2.5.8.4 Example

```
<some_tosca_entity>:
  properties:
    app_latency:
      max_latency: 20000000
```

5.2.5.8.5 Additional Requirements

None.

5.2.5.9 tosca.datatypes.mec.UserContextTransferCapability

5.2.5.9.1 Description

The UserContextTransferCapability data type describes information of user context transfer capability of the application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.9.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.9.1-1: Type name, shorthand and URI

Shorthand Name	UserContextTransferCapability
Type Qualified Name	toscamec:UserContextTransferCapability
Type URI	tosca.datatypes.mec.UserContextTransferCapability

5.2.5.9.2 Properties

The properties of the UserContextTransferCapability data type shall comply with the provisions set out in table 5.2.5.9.2-1.

Table 5.2.5.9.2-1: Properties

Name	Required	Type	Constraints	Description
stateful_application	yes	boolean		If the application is stateful, this attribute shall be set to true. Otherwise, this attribute shall be set to false.
user_context_transfer_support	no	boolean		This attribute shall be present if the application is stateful and shall be absent otherwise. If the application supports the user context transfer, this attribute shall be set to true, otherwise this attribute shall be set to false.

5.2.5.9.3 Definition

The syntax of the UserContextTransferCapability data type shall comply with the following definition:

```
  tosca.datatypes.mec.UserContextTransferCapability:
    derived_from: tosca.datatypes.Root
    description: information of user context transfer capability of the application
    properties:
      stateful_application:
        type: boolean
        description: whether the application is stateful.
        required: true
      user_context_transfer_support:
        type: boolean
        description: whether the application supports the user context transfer capability, only
applicable when the application is stateful
        required: false
```

5.2.5.9.4 Example

```
<some_tosca_entity>:
  properties:
    user_context_transfer_capability:
      stateful_application: true
      user_context_transfer_support: true
```

5.2.5.9.5 Additional Requirements

None.

5.2.5.10 tosca.datatypes.mec.AppNetworkPolicy

5.2.5.10.1 Description

The AppNetworkPolicy data type describes the network policy used in the application instantiation and operation, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.10.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.10.1-1: Type name, shorthand and URI

Shorthand Name	AppNetworkPolicy
Type Qualified Name	toscamec:AppNetworkPolicy
Type URI	tosca.datatypes.mec.AppNetworkPolicy

5.2.5.10.2 Properties

The properties of the AppNetworkPolicy data type shall comply with the provisions set out in table 5.2.5.10.2-1.

Table 5.2.5.10.2-1: Properties

Name	Required	Type	Constraints	Description
cellular_network	no	boolean		If present, and the application prefers to a cellular network to carry its traffic, this attribute shall be set to true. Otherwise, it shall be set to false.
wifi_network	no	boolean		If present, and the application prefers to a Wi-Fi network to carry its traffic, this attribute shall be set to true. Otherwise, it shall be set to false.
fixed_access_network	no	boolean		If present, and the application prefers to a fixed access network to carry its traffic, this attribute shall be set to true. Otherwise, it shall be set to false.

5.2.5.10.3 Definition

The syntax of the AppNetworkPolicy data type shall comply with the following definition:

```
  tosca.datatypes.mec.AppNetworkPolicy:
    derived_from: tosca.datatypes.Root
    description: network policy used in the application instantiation and operation
    properties:
      cellular_network:
        type: boolean
        description: whether the application prefers a cellular network to carry the traffic.
        required: false
      wifi_network:
        type: boolean
        description: whether the application prefers a Wi-Fi network to carry the traffic.
        required: false
      fixed_access_network:
        type: boolean
        description: whether the application prefers a fixed access network to carry the traffic.
        required: false
```

5.2.5.10.4 Example

```
<some_tosca_entity>:
  properties:
    app_network_policy:
      cellular_network: true
      wifi_network: false
      fixed_access_network: false
```

5.2.5.10.5 Additional Requirements

None.

5.2.5.11 tosca.datatypes.mec.CategoryRef

5.2.5.11.1 Description

The CategoryRef data type describes the category reference, as defined in ETSI GS MEC 011 [i.4]. Table 5.2.5.11.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.11.1-1: Type name, shorthand and URI

Shorthand Name	CategoryRef
Type Qualified Name	toscamec:CategoryRef
Type URI	tosca.datatypes.mec.CategoryRef

5.2.5.11.2 Properties

The properties of the CategoryRef data type shall comply with the provisions set out in table 5.2.5.11.2-1.

Table 5.2.5.11.2-1: Properties

Name	Required	Type	Constraints	Description
href	yes	string		Reference of the category.
id	yes	string		Unique identifier of the category.
name	yes	string		Name of the category.
version	yes	string		Category version.

5.2.5.11.3 Definition

The syntax of the CategoryRef data type shall comply with the following definition:

```
  tosca.datatypes.mec.CategoryRef:
    derived_from: tosca.datatypes.Root
    description: a category reference
    properties:
      href:
        type: string
        description: reference of the category.
        required: true
      id:
        type: string
        description: unique identifier of the category.
        required: true
      name:
        type: string
        description: name of the category.
        required: true
      version:
        type: string
        description: category version.
        required: true
```

5.2.5.11.4 Example

```
<some_tosca_entity>:
  properties:
    ser_category:
      href: www.example.com/service_loc
      id: service_loc
      name: Location
      version: '1.0'
```

5.2.5.11.5 Additional Requirements

None.

5.2.5.12 tosca.datatypes.mec.Permission

5.2.5.12.1 Description

The Permission data type describes an access right of a particular MEC service, as defined in clause 7.2 of ETSI GS MEC 009 [5]. Table 5.2.5.12.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.12.1-1: Type name, shorthand and URI

Shorthand Name	Permission
Type Qualified Name	toscamec:Permission
Type URI	tosca.datatypes.mec.Permission

5.2.5.12.2 Properties

The properties of the Permission data type shall comply with the provisions set out in table 5.2.5.12.2-1.

Table 5.2.5.12.2-1: Properties

Name	Required	Type	Constraints	Description
id	yes	string		A string that identifies the item to which access is granted or denied (for example read access or read/write access to a resource or a group of resources in a REST API). It is unique within the scope of a particular MEC service specification. See note.
display_name	no	string		A short human-readable string to describe the permission when represented towards human users.
NOTE: The permitted values defined by the respective service. For MEC defined services, table 5.2.5.2.2-2 provides the references to the document defining the permissions.				

5.2.5.12.3 Definition

The syntax of the Permission data type shall comply with the following definition:

```
  tosca.datatypes.mec.Permission:
    derived_from: tosca.datatypes.Root
    description: access right of a particular MEC service
    properties:
      id:
        type: string
        description: name of a right within a particular MEC service
        required: true
      display_name:
        type: string
        description: human-readable string to describe the permission
        required: false
```

5.2.5.12.4 Example

```
<some_tosca_entity>:
  properties:
    requested_permissions:
      - id: queries
        display_name: Queries
```

5.2.5.12.5 Additional Requirements

None.

5.2.5.13 tosca.datatypes.mec.TransportSerializerDescriptor

5.2.5.13.1 Description

The TransportSerializerDescriptor data type describes a transport, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.13.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.13.1-1: Type name, shorthand and URI

Shorthand Name	TransportSerializerDescriptor
Type Qualified Name	toscamec:TransportSerializerDescriptor
Type URI	tosca.datatypes.mec.TransportSerializerDescriptor

5.2.5.13.2 Properties

The properties of the TransportSerializerDescriptor data type shall comply with the provisions set out in table 5.2.5.4.2-1.

Table 5.2.5.13.2-1: Properties

Name	Required	Type	Constraints	Description
transport	yes	tosca.datatypes.mec.TransportDescriptor		Information about the transport in this transport binding.
serializers	yes	list of string	valid value: [JSON, XML, PROTOBUF3]	Information about the serializers in this transport binding. Support for at least one of the entries is required in conjunction with the transport.

5.2.5.13.3 Definition

The syntax of the TransportSerializerDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.TransportSerializerDescriptor:
    derived_from: tosca.datatypes.Root
    description: requirements of a MEC application to the transport bindings
    properties:
      transport:
        type: tosca.datatypes.mec.TransportDescriptor
        description: information about the transport in this transport binding
        required: true
      serializers:
        type: list
        description: information about the serializers in this transport binding
        required: true
        entry_schema:
          type: string
          constraints:
            - valid_values: [ JSON, XML, PROTOBUF3 ]
```

5.2.5.13.4 Example

```
<some_tosca_entity>:
  properties:
    transport_supported:
      - transport:
          type: REST_HTTP
          protocol: HTTP
          version: '2.0'
          security:
            …
        serializers:
          - JSON
```

5.2.5.13.5 Additional Requirements

None.

5.2.5.14 tosca.datatypes.mec.TransportDescriptor

5.2.5.14.1 Description

The TransportDescriptor data type describes a transport, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.14.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.14.1-1: Type name, shorthand and URI

Shorthand Name	TransportDescriptor
Type Qualified Name	toscamec:TransportDescriptor
Type URI	tosca.datatypes.mec.TransportDescriptor

5.2.5.14.2 Properties

The properties of the TransportDescriptor data type shall comply with the provisions set out in table 5.2.5.14.2-1.

Table 5.2.5.14.2-1: Properties

Name	Required	Type	Constraints	Description
type	yes	string	valid_values: [REST_HTTP, MB_TOPIC_BASED, MB_ROUTING, MB_PUBSUB, RPC, RPC_STREAMING, WEBSOCKET]	Type of the transport.
protocol	yes	string		The name of the protocol used. Shall be set to "HTTP" for a REST API.
version	yes	string		The version of the protocol used.
security	yes	tosca.datatypes.mec.SecurityInfo		Information about the security used by the transport.

5.2.5.14.3 Definition

The syntax of the TransportDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.TransportDescriptor:
    derived_from: tosca.datatypes.Root
    description: a transport
    properties:
      type:
        type: string
        description: type of the transport.
        required: true
        constraints:
          - valid_values: [ REST_HTTP, MB_TOPIC_BASED, MB_ROUTING, MB_PUBSUB, RPC, RPC_STREAMING,
WEBSOCKET ]
      protocol:
        type: string
        description: name of the protocol used.
        required: true
      version:
        type: string
        description: version of the protocol used.
        required: true
      security:
        type: tosca.datatypes.mec.SecurityInfo
        description: information about the security used by the transport.
        required: true
```

5.2.5.14.4 Example

```
<some_tosca_entity>:
  properties:
    transport:
      type: REST_HTTP
      protocol: HTTP
      version: '2.0'
      security:
        …
```

5.2.5.14.5 Additional Requirements

None.

5.2.5.15 tosca.datatypes.mec.SecurityInfo

5.2.5.15.1 Description

The SecurityInfo data type describes security information related to a transport, as defined in ETSI GS MEC 011 [i.4]. Table 5.2.5.15.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.15.1-1: Type name, shorthand and URI

Shorthand Name	SecurityInfo
Type Qualified Name	toscamec:SecurityInfo
Type URI	tosca.datatypes.mec.SecurityInfo

5.2.5.15.2 Properties

The properties of the SecurityInfo data type shall comply with the provisions set out in table 5.2.5.15.2-1.

Table 5.2.5.15.2-1: Properties

Name	Required	Type	Constraints	Description
oauth2_grant_type	no	list of string	valid_values: [OAUTH2_AUTHORIZATION_CODE, OAUTH2_IMPLICIT_GRANT, OAUTH2_RESOURCE_OWNER, OAUTH2_CLIENT_CREDENTIALS]	Supported OAuth 2.0 grant types.
oauth2_token_endpoint	no	string		The token endpoint. Shall be present unless the grant type is OAUTH2_IMPLICIT_GRANT.

5.2.5.15.3 Definition

The syntax of the SecurityInfo data type shall comply with the following definition:

```
  tosca.datatypes.mec.SecurityInfo:
    derived_from: tosca.datatypes.Root
    description: security information related to a transport
    properties:
      oauth2_grant_type:
        type: list
        description: supported OAuth 2.0 grant types.
        required: false
        entry_schema:
          type: string
          constraints:
            - valid_values: [ OAUTH2_AUTHORIZATION_CODE, OAUTH2_IMPLICIT_GRANT,
OAUTH2_RESOURCE_OWNER, OAUTH2_CLIENT_CREDENTIALS ]
      oauth2_token_endpoint:
        type: string
        description: token endpoint URI.
        required: false
```

5.2.5.15.4 Example

```
<some_tosca_entity>:
  properties:
    security:
      oauth2_grant_type: OAUTH2_CLIENT_CREDENTIALS
      oauth2_token_endpoint: www.example.com/token
```

5.2.5.15.5 Additional Requirements

None.

5.2.5.16 tosca.datatypes.mec.TrafficFilter

5.2.5.16.1 Description

The TrafficFilter data type describes the specification of MEC application requirements related to traffic rules, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.16.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.16.1-1: Type name, shorthand and URI

Shorthand Name	TrafficFilter
Type Qualified Name	toscamec:TrafficFilter
Type URI	tosca.datatypes.mec.TrafficFilter

5.2.5.16.2 Properties

The properties of the TrafficFilter data type shall comply with the provisions set out in table 5.2.5.16.2-1.

Table 5.2.5.16.2-1: Properties

Name	Required	Type	Constraints	Description
src_address	no	list of string		An IP address or a range of IP addresses. For IPv4, the IP address could be an IP address plus mask, or an individual IP address, or a range of IP addresses. For IPv6, the IP address could be an IP prefix, or a range of IP prefixes.
dst_address	no	list of string		An IP address or a range of IP addresses. For IPv4, the IP address could be an IP address plus mask, or an individual IP address, or a range of IP addresses. For IPv6, the IP address could be an IP prefix, or a range of IP prefixes.
src_port	no	list of string		A port or a range of ports.
dst_port	no	list of string		A port or a range of ports.
protocol	no	list of string		Specify the protocol of the traffic filter.
tag	no	list of string		Used for tag based traffic rule.
src_tunnel_address	no	list of string		Used for GTP tunnel based traffic rule.
tgt_tunnel_address	no	list of string		Used for GTP tunnel based traffic rule.
src_tunnel_port	no	list of string		Used for GTP tunnel based traffic rule.
dst_tunnel_port	no	list of string		Used for GTP tunnel based traffic rule.
qci	no	integer	greater_than: 0	Used to match all packets that have the same QCI.
dscp	no	integer		Used to match all IPv4 packets that have the same DSCP.
tc	no	integer		Used to match all IPv6 packets that have the same TC.

5.2.5.16.3 Definition

The syntax of the TrafficFilter data type shall comply with the following definition:

```
tosca.datatypes.mec.TrafficFilter:
  derived_from: tosca.datatypes.Root
  description: specification of MEC application requirements related to traffic rules
  properties:
    src_address:
      type: list
      description: source IP address(es).
      required: false
```

```
          entry_schema:
            type: string
        dst_address:
          type: list
          description: destination IP address(es).
          required: false
          entry_schema:
            type: string
        src_port:
          type: list
          description: source port(s).
          required: false
          entry_schema:
            type: string
        dst_port:
          type: list
          description: destination port(s).
          required: false
          entry_schema:
            type: string
        protocol:
          type: list
          description: protocol of the traffic filter.
          required: false
          entry_schema:
            type: string
        tag:
          type: list
          description: used for tag based traffic rule.
          required: false
          entry_schema:
            type: string
        src_tunnel_address:
          type: list
          description: source IP address(es) for GTP tunnel.
          required: false
          entry_schema:
            type: string
        tgt_tunnel_address:
          type: list
          description: target IP address(es) for GTP tunnel.
          required: false
          entry_schema:
            type: string
        src_tunnel_port:
          type: list
          description: source port(s) for GTP tunnel.
          required: false
          entry_schema:
            type: string
        dst_tunnel_port:
          type: list
          description: destination port(s) for GTP tunnel.
          required: false
          entry_schema:
            type: string
        qci:
          type: integer
          description: QCI of the packets.
          required: false
          constraints:
            - greater_than: 0
        dscp:
          type: integer
          description: DSCP of IPv4 packets.
          required: false
        tc:
          type: integer
          description: TC of IPv6 packets.
          required: false
```

5.2.5.16.4 Example

```
<some_tosca_entity>:
  properties:
    traffic filter:
      - src_address:
- 192.168.1.1
        dst_address:
          - 192.168.1.2
        src_port:
          - 8080
        dst_port:
          - 8080
        protocol:
          - udp
        tag:
          - 1234
        src_tunnel_address:
          - 10.10.10.10
        tgt_tunnel_address:
          - 10.10.10.20
        src_tunnel_port:
          - 8080
        dst_tunnel_port:
          - 8080
        qci: 1
        dscp: 0
        tc: 0
```

5.2.5.16.5 Additional Requirements

None.

5.2.5.17 tosca.datatypes.mec.InterfaceDescriptor

5.2.5.17.1 Description

The InterfaceDescriptor data type describes an interface of a MEC application, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.17.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.17.1-1: Type name, shorthand and URI

Shorthand Name	InterfaceDescriptor
Type Qualified Name	toscamec:InterfaceDescriptor
Type URI	tosca.datatypes.mec.InterfaceDescriptor

5.2.5.17.2 Properties

The properties of the InterfaceDescriptor data type shall comply with the provisions set out in table 5.2.5.17.2-1.

Table 5.2.5.17.2-1: Properties

Name	Required	Type	Constraints	Description
interface_type	yes	string	valid_values: [TUNNEL, MAC, IP]	Type of interface.
tunnel_info	no	tosca.datatypes.mec.TunnelInfo		Included only if the destination address type is tunnel.
src_mac_address	no	string		If the interface type is MAC, the source address identifies the MAC address of the interface.
dst_mac_address	no	string		If the interface type is MAC, the destination address identifies the MAC address of the destination.
dst_ip_address	no	string		If the interface type is IP, the destination address identifies the IP address of the destination.

5.2.5.17.3 Definition

The syntax of the InterfaceDescriptor data type shall comply with the following definition:

```
  tosca.datatypes.mec.InterfaceDescriptor:
    derived_from: tosca.datatypes.Root
    description: an interface of a MEC application
    properties:
      interface_type:
        type: string
        description: type of interface.
        required: true
        constraints:
          - valid_values: [ TUNNEL, MAC, IP ]
      tunnel_info:
        type: tosca.datatypes.mec.TunnelInfo
        description: information of the tunnel, if the interface type is TUNNEL.
        required: false
      src_mac_address:
        type: string
        description: source address of the interface, if the interface type is MAC.
        required: false
      dst_mac_address:
        type: string
        description: destination address of the interface, if the interface type is MAC.
        required: false
      dst_ip_address:
        type: string
        description: destination address of the interface, if the interface type is IP.
        required: false
```

5.2.5.17.4 Example

```
<some_tosca_entity>:
  properties:
    dst_interface:
      interface_type: TUNNEL
      tunnel_info:
        …
      src_mac_address: 02-00-00-00-00-00
      dst_mac_address: 02-00-00-00-00-01
      dst_ip_address: 192.0.2.1
```

5.2.5.17.5 Additional Requirements

None.

5.2.5.18 tosca.datatypes.mec.TunnelInfo

5.2.5.18.1 Description

The TunnelInfo data type describes the information about a tunnel transport, as defined in ETSI GS MEC 010-2 [1]. Table 5.2.5.18.1-1 specifies the declared names for this data type. These names shall be used as specified in TOSCA-Simple-Profile-YAML-v1.3 [2].

Table 5.2.5.18.1-1: Type name, shorthand and URI

Shorthand Name	TunnelInfo
Type Qualified Name	toscamec:TunnelInfo
Type URI	tosca.datatypes.mec.TunnelInfo

5.2.5.18.2 Properties

The properties of the TunnelInfo data type shall comply with the provisions set out in table 5.2.5.18.2-1.

Table 5.2.5.18.2-1: Properties

Name	Required	Type	Constraints	Description
tunnel_type	yes	string	valid_values: [GTP-U, GRE]	Type of tunnel.
tunnel_dst_address	yes	string		Destination address of the tunnel.
tunnel_src_address	yes	string		Source address of the tunnel.
tunnel_specific_data	no	string		Parameters specific to the tunnel.

5.2.5.18.3 Definition

The syntax of the TunnelInfo data type shall comply with the following definition:

```
  tosca.datatypes.mec.TunnelInfo:
    derived_from: tosca.datatypes.Root
    description: information about a tunnel transport.
    properties:
      tunnel_type:
        type: string
        description: type of tunnel.
        required: true
        constraints:
          - valid_values: [ GTP-U, GRE ]
      tunnel_dst_address:
        type: string
        description: destination address of the tunnel.
        required: true
      tunnel_src_address:
        type: string
        description: source address of the tunnel.
        required: true
      tunnel_specific_data:
        type: string
        description: parameters specific to the tunnel.
        required: false
```

5.2.5.18.4 Example

```
<some_tosca_entity>:
  properties:
    tunnel_info:
      tunnel_type: GTP-U
      tunnel_dst_address: 10.10.10.10
      tunnel_src_address: 11.11.11.11
      tunnel_specific_data: xxx
```

5.2.5.18.5 Additional Requirements

None.

5.2.6 Capability type definitions

The MEC application type definitions derive capability types from NFV by inheriting from the tosca.nodes.nfv.VNFnode type, but do not define additional capability types.

5.2.7 Relationship type definitions

The MEC application type definitions derive relationship types from NFV by inheriting from the tosca.nodes.nfv.VNF node type, but do not define additional relationship types.

5.2.8 Policy type definitions

The MEC system does not make use of policy types. But to allow MEC in NFV deployments, a statement defining the instantiation level is required as defined in ETSI GS NFV-SOL 001 [3] and shall be added to every AppD service template. It shall define a single instantiation level with the number of instances set to one, as shown in the example.

EXAMPLE:

```
policies:
  - instantiation_levels:
      type: tosca.policies.nfv.InstantiationLevels
      properties:
        levels:
          mec_instantiation_level:
            description: 'mec instantiation level'

  - mymecapp_instantiation_level:
      type: tosca.policies.nfv.VduInstantiationLevels
      properties:
        levels:
          mec_instantiation_level:
            number_of_instances: 1 # always set to 1
      targets: [ MecApp ]
```

5.2.9 Artifact type definitions

The MEC application type definitions use the tosca.artifacts.nfv.SwImage from NFV by using the tosca.nodes.nfv.Vdu.Compute node type. No other artifacts are required.

5.2.10 Interface type definitions

The MEC application type definitions derive interface types from NFV by inheriting from the tosca.nodes.nfv.VNF node type, but do not define additional interface types.

5.2.11 Group type definitions

The MEC system does not make use of group types.

5.3 MEC application descriptor based on YANG specification

The format to specify the AppD in YANG is out of scope of the present document.

6 MEC application package

6.1 MEC application package structure and format

MEC application package is a TOSCA YAML CSAR file which is an archive file using the ZIP file format which shall conform to Document Container Format File [5]. The structure and format of a MEC application package shall conform to the CSAR format with a TOSCA-Metadata directory defined in ETSI GS NFV-SOL 004 [4], clause 4.1. The CSAR file may optionally contain other directories with bespoke names and contents.

In addition, for supporting on-boarded on NFVO, a MEC application package shall be a valid VNF package.

6.2 MEC application package file contents

6.2.1 General

A MEC application package shall contain a main TOSCA definitions YAML file representing all or part of the AppD, and additional files.

Examples of MEC application package options are the same as the VNF package examples given in annex A of ETSI GS NFV-SOL 004 [4]. Examples related to the manifest file are in clause 6.2.2 of the present document.

6.2.2 MEC application package manifest file

A CSAR MEC application package shall have a manifest file. 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.

The manifest file shall start with the MEC application 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 6.2.2-1 and the values shall comply with the provisions specified in table 6.2.2-1. Table 6.2.2-1 illustrates the name-value pairs in the manifest file. Because the application package is a valid VNF package, the "name" of some name-value pairs shall be the same as they are in the manifest file of a VNF package.

The "required" column in table 6.2.2-1 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 6.2.2-1: MEC application package metadata

Name	Value	Required	Description
vnfd_id	A sequence of UTF-8 characters. See note 1.	Yes	Refer to appDId in table 6.2.1.2.2-1 in ETSI GS MEC 010-2 [1].
vnf_provider_id	A sequence of UTF-8 characters. See note 1.	Yes	Refer to appProvider in table 6.2.1.2.2-1 in ETSI GS MEC 010-2 [1].
vnf_product_name	A sequence of UTF-8 characters. See note 1.	Yes	Refer to appName in table 6.2.1.2.2-1 in ETSI GS MEC 010-2 [1].
vnf_release_date_time	A string formatted according to IETF RFC 3339 [6]	Yes	The release date time of the application.
vnf_software_version	A string. See note 1.	Yes	Refer to appSoftVersion in table 6.2.1.2.2-1 in ETSI GS MEC 010-2 [1].
vnf_package_version	A string. See note 2.	Yes	Refer to appDVersion in table 6.2.1.2.2-1 in ETSI GS MEC 010-2 [1].
compatible_specification_versions	See note 5.	Yes	Indicates which versions of the ETSI GS NFV-SOL 004 [4] the application package complies to, as known at package creation time. See notes 3 and 4.
mec_versions	See note 5.	Yes	Indicates which versions of the present document the application package complies to, as known at package creation time. See note 3.
vnfm_info	A comma-separated list of strings as defined in the VNFD. Whitespace between list entries shall be trimmed before validation.	Yes	Set to the default value "etsivnfm:v2.3.1" unless a VNF-specific VNFM is required to manage the lifecycle of the MEC app.
app_info_name	A string.	No	Human readable name for the MEC application.
app_description	A string.	No	Human readable description of the MEC application.
NOTE 1: The value shall be identical to those specified in the AppD. NOTE 2: The value shall be identical to the vnfdVersion attribute specified in the AppD. 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 ETSI GS NFV-SOL 004 [4] 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 ETSI GS NFV-SOL 004 [4], i.e. a version prior to 2.7.1 and the package shall be considered valid. NOTE 5: 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. Whitespace between list entries shall be trimmed before validation.			

An example of valid manifest file metadata entries follows.

EXAMPLE 1:

```
metadata:
vnfd_id: 2116fd24-83f2-416b-bf3c-ca1964793aca
vnf_provider_id: MyCompany
vnf_product_name: mec-application
vnf_software_version: 1.0.0
vnf_package_version: 1.0
vnf_release_date_time: 2021-01-01T10:00:00+03:00
vnfm_info: etsivnfm:v2.3.1
compatible_specification_versions: 1.1.1
mec_versions: 3.1.1
```

END OF EXAMPLE 1.

The manifest file shall include a list of all files contained in or referenced from the MEC application 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 MEC application package when authenticity and integrity of the MEC application package is implemented according to option 1 as specified in ETSI GS NFV-SOL 004 [4], clause 5.1.

EXAMPLE 2:

```
Source: application.yaml
Algorithm: SHA-256
Hash: 09e5a788acb180162c51679ae4c998039fa6644505db2415e35107d1ee213943

Source: scripts/install.sh
Algorithm: SHA-256
Hash: d0e7828293355a07c2dccaaa765c80b507e60e6167067c950dc2e6b0da0dbd8b

Source: https://www.vendor_org.com/application/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 36f945953929812aca2701b114b068c71bd8c95ceb3609711428c26325649165
```

END OF EXAMPLE 2.

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

If the MEC application package does not refer to external files, the manifest files may contain digests of individual files contained in the package. If the manifest file does not include digests, the complete CSAR file shall be digitally signed by the VNF provider. A consumer of the MEC application 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, or alternatively, the signature of the CSAR file, is the key for decision regarding a MEC application 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 ETSI GS NFV-SOL 004 [4], clause 5. The VNF refers to MEC application, the VNFD refers to AppD in MEC if they appear in ETSI GS NFV-SOL 004 [4].

6.2.3 Certificate file

The CSAR MEC application 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, 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.

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 MEC application provider, 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.

6.2.4 Non-MANO artifact sets in a MEC application package

ETSI GS NFV-SOL 001 [3] defines non-MANO artifacts and artifact sets for use by functional blocks beyond NFV-MANO. Thus non-MANO artifacts and artifact sets can be included in MEC application packages and be used by the MEC applications in an application specific way.

The use of non-MANO artifacts is not required to provide a valid MEC application package.

6.3 Adding security to MEC application package

The detailed requirements and examples are described in ETSI GS NFV-SOL 004 [4], clause 5, in which the VNF refers to MEC application and the VNFD refers to AppD in MEC.

Annex A (informative): Example for AppD

This example illustrates the TOSCA definitions for a MEC application. It uses some of the examples in clause 5 of the present document and examples from ETSI GS NFV-SOL 001 [3].

In the example the parts derived from NFV are written in blue, parts defined in the present document are written in green, while parts provided by the MEC application provider are written in black.

In a first file, the specific node type definition is created in **mymecappsimple.appdtype.tosca.yaml**. It sets the constraints and defaults required by ETSI GS NFV-SOL 001 [3], clause 6.8.1.8 and defines application specific attributes if needed.

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: Node type for simple MEC application example

imports:
  - https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml

node_types:
  MyCompany.MyMecApp_1.0:
    derived_from: tosca.nodes.mec.MecApp
    properties:
      descriptor_id:
        type: string
        constraints: [ equal: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177 ]
        default: b1bb0ce7-ebca-4fa7-95ed-4840d70a1177
      provider:
        type: string
        constraints: [ equal: MyCompany ]
        default: MyCompany
      product_name:
        type: string
        constraints: [ equal: MyMecApp ]
        default: MyMecApp
      software_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      descriptor_version:
        type: string
        constraints: [ equal: '1.0' ]
        default: '1.0'
      vnfm_info:
        type: list
        entry_schema:
          type: string
        constraints: [ equal: [ 'etsivnfm:v2.3.1' ] ]
        default: [ 'etsivnfm:v2.3.1' ]
      mymecapp_definitions:
        type: list
        entry_schema:
          type: string
        default: ""
```

In this example the AppD service template is provided in a separate yaml file.

```
tosca_definitions_version: tosca_simple_yaml_1_3

description: Simple MEC application example

imports:
  - path/mymecappsimple.appdtype.tosca.yaml

topology template:
  substitution_mappings:
    node_type: MyCompany.MyMecApp_1.0
    requirements:
      virtual_link: [ MyMecAppCp, virtual_link ]

  node_templates:
    MyMecApp:
      type: MyCompany.MyMecApp_1.0:
      properties:
        mec_version: [ '2.2.1' ]
        app_service_required:
          - ser_name: RNIS
            version: '1.0'
        app_service_produced:
          - ser_name: AMS
            version: '1.0'
            transports_supported:
              - transport:
                  type: REST_HTTP
                  protocol: HTTP
                  version: '2.0'
                  security:
                    oauth2_grant_type: [ OAUTH2_CLIENT_CREDENTIALS ]
                    oauth2_token_endpoint: 'www.example.com/token'
                serializers:
                  - JSON
        app_feature_required:
          - feature_name: UserApps
            version: '1.0'
        transport_dependencies:
          - transport_serializer:
              transport:
                type: REST_HTTP
                protocol: HTTP
                version: '2.0'
                security:
                  oauth2_grant_type: [ OAUTH2_CLIENT_CREDENTIALS ]
                  oauth2_token_endpoint: '/token'
              serializers:
                - JSON
            labels:
              - test
        app_traffic_rule:
          - traffic_rule_id: rule1
            filter_type: FLOW
            priority: 1
            traffic_filter:
              - src_address:
                  - 192.168.1.1
                dst_address:
                  - 192.168.1.2
                src_port:
                  - 8080
                dst_port:
                  - 8080
                protocol:
                  - udp
                tag:
                  - 1234
                src_tunnel_address:
                  - 10.10.10.1
                tgt_tunnel_address:
                  - 10.10.10.2
                src_tunnel_port:
                  - 8080
                dst_tunnel_port:
                  - 8080
                qci: 1
                dscp: 0
                tc: 1
            action: DROP
            dst_interface:
              - interface_type: TUNNEL
                tunnel_info:
                  tunnel_type: GTP-U
                  tunnel_dst_address: 10.10.10.2
                  tunnel_src_address: 10.10.10.1
              src_mac_address: 02-00-00-00-00-00
              dst_mac_address: 02-00-00-00-00-01
              dst_ip_address: 192.0.2.1
        app_dns_rule:
          - dns_rule_id: dnsrule1
            domain_name: www.example.com
            ip_address_type: IP_V4
            ip_address: 192.0.2.1
          ttl: 100
        app_latency:
          max_latency: 20000000
        user_context_transfer_capability:
          stateful_application: true
          user_context_transfer_support: true
        app_network_policy:
          cellular_network: true
          wifi_network: false
          fixed_access_network: false
        lcm_operations_configuration: # optional in NFV, used for terminateAppInstanceOpConfig
          terminate: # other attributes not relevant for MEC
            min_graceful_termination_timeout: 60 s
            max_recommended_graceful_termination_timeout: 600 s
          operate: # other attributes not relevant for MEC
            min_graceful_stop_timeout: 60 s
            max_recommended_graceful_stop_timeout: 600 s

    MyMecAppNode:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: MyMecApp
        description: Compute node for MyMecApp
        nfvi_constraints:
          key_1: value_1
          key_2: value_2
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 1
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 8192 MiB
            virtual_cpu:
              cpu_architecture: x86
              num_virtual_cpu: 2
              virtual_cpu_clock: 1800 MHz
      requirements:
        - virtual_storage: MyMecAppStorage

    MyMecAppStorage:
      type: tosca.nodes.nfv.Vdu.VirtualBlockStorage
      properties:
        virtual_block_storage_data:
          size_of_storage: 100 GB
          rdma_enabled: true

      artifacts:
        sw_image:
          type: tosca.artifacts.nfv.SwImage
          file: mymecapp.image.v1.0.qcow2
          properties:
            name: MyMecApp Software
            version: '1.0'
            checksum:
              algorithm: sha-256
              hash: b9c3036539fd7a5f87a1bf38eb05fdde8b556a1a7e664dbeda90ed3cd74b4f9d
            container_format: bare
            disk_format: qcow2
            min_disk: 2 GB
            min_ram: 8192 MiB
            size: 2 GB
            operating_system: Linux
            supported_virtualisation_environments:
              - KVM

    MyMecAppCp:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv4 ]
        role: leaf
        description: External connection point to access MyMecApp on IPv4
        protocol: [ associated_layer_protocol: ipv4 ]
        trunk_mode: false
      requirements:
        - virtual_binding: MyMecAppNode
```

```
  policies:
    - instantiation_levels:
        type: tosca.policies.nfv.InstantiationLevels
        properties:
          levels:
            mec_instantiation_level: #user could choose different name
              description: 'mec instantiation level'

    - mymecapp_instantiation_level: #user could choose different name
        type: tosca.policies.nfv.VduInstantiationLevels
        properties:
          levels:
            mec_instantiation_level:
              number_of_instances: 1 # always set to 1
        targets: [ MyMecAppNode ] # needs to correspond to the tosca.nodes.nfv.Vdu.Compute above
```

Annex B (normative): Type definition files

All type definitions specified in in clause 5 of the present document are provided in one single definition file which is available at the following URL:

- https://forge.etsi.org/rep/mec/gs037-pkg-format/raw/v3.1.1/MecAppDTypes.yaml

NOTE 1: The file MecAppDTypes.yaml includes a TOSCA import definition referencing etsi_nfv_sol001_vnfd_types.yaml file. If the later file is included in the MEC application package, the import definition can reference the local file using appropriate path in the MEC application package.

This file is a TOSCA service template that only contains definitions. The template_version in the metadata section within this template is structured as x.y.z, where x, y and z represent the version of this file and are set respectively to "3", "1" and "1" for this version of the present document. In subsequent versions of the present document, "x", "y" and "z" in the template_version will be incremented only if there are changes in the AppD type definitions.

A TOSCA service template representing a MEC application complying with the present document shall contain import statement referencing this file, as defined in clause 5.2.3.2.

NOTE 2: This file may, but need not, be included in the MEC application Package.

Annex C (normative): Conformance

C.1 Purpose

The present document specifies a data model for the AppD by using the grammar defined in the TOSCA-Simple-Profile-YAML-v1.3 [2]. This annex specifies the requirements to be fulfilled for claiming conformance to the present document.

C.2 ETSI MEC TOSCA YAML service template

An AppD conforms to the present document if it complies with all the requirements below:

1) An AppD conformant to the present document shall comply with the requirements in clause 5 of the present document and to the specification of the elements of the TOSCA-Simple-Profile-YAML-v1.3 [2] it uses, unless otherwise stated in clause 5 of the present document.

2) When using or referring to the TOSCA normative types for the VNFD listed in table C.2-1 of ETSI GS NFV-SOL 001 [3], it is valid according to the definitions given in clause 6 of ETSI GS NFV-SOL 001 [3] and to section 5 of the TOSCA-Simple-Profile-YAML-v1.3 [2].

3) When using or referring to the NFV TOSCA normative types listed in table C.2-1, it is valid according to the definitions given in clause 5 of the present document, to clause 6 of ETSI GS NFV-SOL 001 [3] and to section 5 of the TOSCA-Simple-Profile-YAML-v1.3 [2].

Table C.2-1: NFV TOSCA normative types used in the present document

Types
tosca.nodes.nfv.VNF
tosca.nodes.nfv.Vdu.Compute
tosca.artifacts.nfv.SwImage
tosca.nodes.nfv.Vdu.VirtualBlockStorage, tosca.nodes.nfv.Vdu.VirtualObjectStorage, tosca.nodes.nfv.Vdu.VirtualFileStorage
tosca.nodes.nfv.VduCp
tosca.datatypes.nfv.VnfTerminateOperationConfiguration
tosca.datatypes.nfv.VnfOperateOperationConfiguration

4) An AppD conformant to the present document shall comply with AppD TOSCA service template design specified in clause 5.2.3 of the present document.

5) An AppD conformant to the present document shall comply with VNFD TOSCA service template design for a VNF with only one deployment flavour specified in clause 6.11.3 of. ETSI GS NFV-SOL 001 [3].

C.3 TOSCA processor

A processor or program conforms to the present document as MEC TOSCA processor for AppD, if it complies with all the requirements below:

1) It can parse and recognize the elements of any AppD that conforms to the present document, and shall generate errors for those documents that fail to conform to the present document.

2) It shall comply with all requirements and implement the semantics associated with the definitions specified in clauses 5 of the present document.

3) It shall resolve the import definitions, as described in clause 5.2.3.2 of the present document.

Annex D (informative): Change History

Date	Version	Information about changes
September 2020	0.0.1	Initial skeleton
May 2021	0.0.2	MEC(20)000377r3 MEC037 Clause 5 draft MEC(21)000186r1 MEC037 Clause 4 Introduction MEC(21)000187 MEC037 Changes to Clause 1 and 3 MEC(21)000188r1 MEC037 Start on clause 5.2
June 2021	0.0.3	MEC(21)000205r1 MEC037 clause 6 MEC(21)000225 MEC037 Clarify versions of NFV and TOSCA MEC(21)000228r1 MEC037 Skeleton and initial requirements of TOSCA based AppD
September 2021	0.0.4	MEC(21)000278 MEC037 MEC versions in manifest file MEC(21)000334 MEC037 Clause 5.2.4 Additional Node Types
December 2021	0.0.5	MEC(21)000571r2 MEC037 Clause 5.2.4 Basic Node Types MEC(21)000572r1 MEC037 Add info to table 5.2.2-1 MEC(21)000573 MEC037 Clause 5.2.5 Data Type Definitions step 1
January 2022	0.0.6	MEC(22)000004 MEC037 Clause 5.2.5 ServiceDependency and Related Data Type Definitions MEC(22)000012 MEC037 Clause 5.2.5.3 ServiceDescriptor MEC(22)000013 MEC037 Clause 5.2.5.4 FeatureDependency MEC(22)000014 MEC037 Clause 5.2.5.5 TransportDependency MEC(22)000019 MEC037 Clause 5.2.5.7 DNSRuleDescriptor MEC(22)000020 MEC037 Clause 5.2.5.8 LatencyDescriptor MEC(22)000021r1 MEC037 Clause 5.2.5.9 UserContextTransferCapability
February 2022	0.0.7	MEC(22)000054r1 MEC037 bug fixing on DNS ttl MEC(22)000055 MEC037 Clause Clause 5.2.5.10 AppNetworkPolicy MEC(22)000018r1 MEC037 Clause 5.2.5.6 TrafficRuleDescriptor
April 2022	0.0.8	MEC(22)000151 MEC037 Fix some syntax errors MEC(22)000162r1 MEC037 Last 2 missing datatypes MEC(22)000164r1 MEC037 Statement on non-MANO artifacts
July 2022	0.0.9	MEC(22)000338 MEC037 Move document to option 2
August 2022	0.0.10	MEC(22)000357r1 MEC037 Resolve some editors notes MEC(22)000369 MEC037 Fix Figure 5.2.3.1-1 MEC(22)000376 MEC037 Clause 5.2.3.2 Examples Import MEC(22)000370 MEC037 Skeleton for Annex MEC(22)000378r1 MEC037 Annex C
October 2022	0.0.11	MEC(22)000415 MEC037 Resolve editors note in clause 5.2.4.1.8 MEC(22)000416r1 MEC037 Updates about MEC010-2 References MEC(22)000417r3 MEC037 Provide AppD Example MEC(22)000427r1 MEC037 Fix clauses 5.2.8, 5.2.9, add 5.2.11 MEC(22)000428r1 MEC037 Clarify package examples MEC(22)000429r1 MEC037 Clarify ser_name and permissions MEC(22)000470r1 MEC037 delete specification property of tosca.datatypes.mec.Permission Corrections of formatting and other editorial fixes
November 2022	0.0.12	MEC(22)000499r3 MEC037 Clarify Instatiation_Level Policy MEC(22)000500r1 MEC037 Move some defaults MEC(22)000504r1 MEC037 Correct Import and an Example MEC(22)000522r1 MEC037 fixing grammar type MEC(22)000530 MEC037 Create Annex B MEC(22)000532 MEC037 create MecAppDTypes in etsi forge (no change in document)
December 2022	0.1.0	MEC(22)000546 MEC037 bug fixing MEC(22)000553r1 MEC037 Prepare stable draft Reached stable draft
December 2022	3.0.0	Stable draft after editHelp review
December 2022	3.0.1	Final draft ready for MEC approval
January 2023	3.0.2	MEC(23)000019r2 MEC037 Nokia review comments MEC(23)000020 MEC037 Resolve Comments from RC MEC(23)000022 MEC037 Correct link to yaml file

History

Document history		
V3.1.1	March 2023	Publication