ETSI GS NFV-SOL 018 V4.5.1 (2023-12)

Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; Profiling specification of protocol and data model solutions for OS Container management and orchestration

Disclaimer

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

Reference
RGS/NFV-SOL018ed451

Keywords
container, data models, NFV, orchestration,
protocol, service, virtualisation

ETSI

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

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

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

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document provides a mapping of the NFV object model for OS container management and orchestration to managed objects of Kubernetes® and Helm™ as specified by the CNCF® along with a specification of a mapping between a common set of input parameters (e.g. derived from VNFD/NSD and/or NFV-MANO RESTful APIs) and output parameters associated to the management and orchestration of the referred managed objects. It profiles the reference Kubernetes® API as NFV protocol and data model solution for OS container management and orchestration, for CIS MCCO management and for CIS Instance management. It profiles the reference Helm™ documentation as NFV protocol and data model solution for management of OS container workload based on an MCIOP. It profiles the reference OCI™ Distribution Specification API (which is based on the Docker™ registry API) as NFV protocol and data model solution for OS container image management. The latest published versions of the reference Kubernetes® API, Helm™ documentation and OCI™ Distribution Specification API are profiled against the requirements on the functions and the management service interfaces of the Container Infrastructure Service Management (CISM) and Container Image Registry (CIR) functions as specified in ETSI GS NFV-IFA 040 [2], ETSI GS NFV-IFA 036 [9] and ETSI GS NFV-IFA 010 [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 NFV-IFA 010: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Functional requirements specification".

[2] ETSI GS NFV-IFA 040: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Requirements for service interfaces and object model for OS container management and orchestration specification".

[3] Kubernetes® API v1.27.0.

[4] Helm™ CLI v3.13.0.

[5] OCI™ Distribution Specification v1.0.0.

[6] OCI™ Image Format Specification v1.0.1.

[7] Helm™ charts v3.13.0.

[8] Kubernetes® reference documentation, API Access Control.

[9] ETSI GS NFV-IFA 036: "Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Requirements for service interfaces and object model for container cluster management and orchestration specification".

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 GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] ETSI GS NFV-SOL 003: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; RESTful protocols specification for the Or-Vnfm Reference Point".

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

[i.4] ETSI NFV Release Documentation.

[i.5] ETSI GS NFV-SOL 013: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

[i.6] Void.

[i.7] ETSI GS NFV-SOL 002: "Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point".

[i.8] Open Container Initiative Charter v1.3.

[i.9] IETF RFC 9110: "HTTP Semantics".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

CIS cluster: set of CIS instances, and one or multiple CISM instances managing them

NOTE: At minimum, the CIS cluster contains one CISM instance and one CIS instance.

CIS cluster enhancement capability: MCCO that provides additional capabilities to a CIS cluster

CIS cluster node: compute resource that runs a Container Infrastructure Service (CIS) instance or a Container Infrastructure Service Management (CISM) instance, or both

NOTE: The CIS cluster node can be either physical (e.g. a bare-metal server), or virtual (e.g. a virtual machine).

Daemon object: MCCO acting as a background process to run in the CISM to deploy MCCO instances having the same functionality onto applicable CIS cluster nodes

managed CIS cluster object: abstract NFV object for CIS cluster management characterized by its configuration, state, requested and allocated infrastructure resources, and applicable operational policies

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR NFV 003 [i.1] and the following apply:

CCEC	CIS Cluster Enhancement Capability
MCCO	Managed CIS Cluster Object

4 Overview of protocols and data models for OS Container management and orchestration

4.1 Summary of ETSI GS NFV-IFA 040 and ETSI GS NFV-IFA 036

ETSI GS NFV-IFA 040 [2] and ETSI GS NFV-IFA 036 [9] specify the requirements on the following services to be provided by the CISM, and the requirements on the management services interfaces to expose these services to other NFV-MANO functional entities and/or external entities outside NFV-MANO:

1) OS container workload management service
2) OS container compute management service
3) OS container storage management service
4) OS container network management service
5) OS container configuration management service
6) CIS instance management service
7) CIS MCCO management service

The OS container workload management service interface is produced by the CISM to enable consumers to request lifecycle management operations on containerized workloads based on a MCIOP and to query information on containerized workloads based on a MCIOP.

The OS container compute/storage/network management service interfaces are produced by the CISM to enable consumers to request management operations on Compute/Storage/Network MCIOs and to query information on Compute/Storage/Network MCIOs.

The OS container configuration management service interface is produced by the CISM to enable consumers to request management operations on MCIO configurations and policies for MCIOs and to query information on those respectively. The OS container configuration management service interface also enables consumers to request management operations on namespaces and namespace quota.

The CIS instance management service interface is produced by the CISM to enable consumers to request lifecycle management operations on CIS instances and to query information on CIS instances.

The CIS MCCO management service interface is produced by the CISM to enable consumers to request management operations on MCCOs and to query information on MCCOs.

ETSI GS NFV-IFA 040 [2] also specifies the requirements on the OS container image management service to be provided by the CIR, and the requirements on the OS container image management service interface to be exposed to other NFV-MANO functional entities and/or external entities outside NFV-MANO. The OS container image management service interface is produced by the CIR to enable consumers to request adding and deleting OS container images to/from the CIR and to query information about OS container images in the CIR.

4.2 Profiled protocol and data model solutions

4.2.1 Kubernetes® API

4.2.1.1 Introduction

This clause provides an overview over the Kubernetes® API [3] which is profiled to the requirements on the OS container management service interfaces exposed by the CISM as specified in ETSI GS NFV-IFA 040 [2]. The overview covers the high level API structure concerning the grouping of the managed resource objects into resource categories as well as the generic concepts for the data model of the managed resource objects.

4.2.1.2 API structure

The Kubernetes® API [3] managed objects represent a concrete instance of a resource type on the CIS cluster. Kubernetes® leverages standard RESTful terminology to describe the API concepts:

- A **resource type** is the name used in the URL.
- All resource types have a representation in JSON (their object schema) which is called a **kind**.
- A list of instances of a resource type is known as a **collection**.
- A single instance of a resource type is called a **resource,** and also usually represents an **object**.

All resource types are either scoped by the CIS cluster (e.g. `/apis/GROUP/VERSION/*`) or to a namespace (e.g. `/apis/GROUP/VERSION/namespaces/NAMESPACE/*`).

Standard HTTP methods POST, PUT, PATCH, and DELETE support single resources only. These methods with single resource support have no support for submitting multiple resources together in an ordered or unordered list or transaction.

The Kubernetes® API [3] supports read and write operations on the Kubernetes® resource objects via a Kubernetes® API endpoint. Kubernetes® differentiates the following categories of resource objects managed via their APIs:

- **Workloads:** objects used to manage and run OS containers on the CIS cluster
- **Discovery & Loadbalancing:** objects used to inter-connect the workloads into externally accessible, load-balanced services
- **Configuration & Storage:** objects used to inject initialization data into the containerized applications, and to persist data that is external to the OS containers
- **Cluster:** objects define how the CIS cluster itself is configured
- **Metadata:** objects used to configure the behaviour of other resources within the CIS cluster

A mapping of the individual Kubernetes® managed resource objects to the NFV object model is provided in clause 5 of the present document.

4.2.1.3 Data model concepts

The Kubernetes® resource objects are modelled with individual object schemas. All resource objects typically have 3 components:

- **Resource ObjectMeta:** The metadata about the resource object, such as its name, type, API version, annotations, and labels. This schema, which is common to all resource types, contains fields that may be updated both by the external user and the CIS system.
- **ResourceSpec:** Defined by the external user and describes the desired state of the system concerning the resource object. Specified when creating or modifying a resource object is requested.

- **ResourceStatus:** Provided by the CIS system and represents the current state of the system concerning the resource object.

4.2.2 Helm™ CLI

4.2.2.1 Introduction

This clause provides an overview over the Helm™ CLI [4] which is profiled to the requirements on the OS container workload management service interface exposed by the CISM as specified in ETSI GS NFV-IFA 040 [2]. Helm™ is a tool for managing OS container workloads deployed on Kubernetes® CIS clusters based on MCIOPs called Helm™ charts. The overview covers the high level CLI structure concerning the main operations as well as the generic concepts for the data model of the MCIOP and the managed runtime objects.

For Helm™, there are three important concepts:

1) The **Helm™ chart** is a bundle of information necessary to create an instance of an OS container workload deployed on Kubernetes® CIS clusters.

2) The **Helm™ config** contains configuration information that can be merged into a packaged Helm™ chart to create a releasable object.

3) A **Helm™ release** is a running instance of an OS container workload based on a Helm™ chart, combined with a specific config.

4.2.2.2 CLI structure

The Helm™ CLI provides commands for the following, common actions:

- Install OS container workloads based on Helm™ charts
- Get information on existing Helm™ releases and their runtime details
- Upgrade a Helm™ release to a new version of a Helm™ chart
- Roll-back a Helm™ release to a previous Helm™ release version
- Uninstall a Helm™ release of a Helm™ chart, removing all of the resources associated with the last Helm™ release
- Display the status information of a Helm™ release

All Helm™ CLI commands do have the generic synopsis:

```
helm [COMMAND] [RELEASE] [CHART] [flags]
```

Variations and details for the specific syntaxes of the commands are described in the corresponding profiling clauses of the present document.

4.2.2.3 Data model concepts

4.2.2.3.1 Helm™ chart file structure

A Helm™ chart is organized as a collection of files inside of a directory. The directory name is the name of the chart (without versioning information). Inside of this directory, Helm™ will expect a structure that matches this:

```
chartname/
  Chart.yaml                  # A YAML file containing information about the Helm™ chart
  LICENSE                     # A plain text file containing the license for the
                              # Helm™ chart
  README.md                   # A human-readable README file
  values.yaml                 # The default configuration values for this Helm™ chart
  values.schema.json          # A JSON Schema for imposing a structure on the
                              # values.yaml file
  charts/                     # A directory containing any Helm™ charts upon which this
                              # Helm™ chart depends.
```

```
crds/                       # Custom Resource Definitions
templates/                  # A directory of templates that, when combined with values,
                            # will generate valid Kubernetes® manifest files.
templates/NOTES.txt         # A plain text file containing short usage notes
```

Helm™ chart reserves the use of the `charts/`, `crds/`, and `templates/` directories, and of the listed file names. Other files will be left as they are.

NOTE: The intent of this list is to provide an overview of the Helm™ chart structure. It is not intended to convey a mandatory or optional presence of each of the structure elements.

4.2.2.3.2 Helm™ release objects

The Helm™ release object is one of the built-in objects of Helm™. The data model describing the properties of a Helm™ release consists of the following main elements:

- **Name:** the name of the Helm™ release
- **Info:** the deployment dates, status information and notes associated to a Helm™ release
- **Chart:** the Helm™ chart that the Helm™ release is based upon
- **Config:** the set of extra values that have been added to the Helm™ chart
- **Manifest:** the string representation of the rendered Helm™ template
- **Hooks:** all of the hooks declared for this Helm™ release
- **Version:** the revision of the Helm™ release
- **Namespace:** the Kubernetes® namespace of the Helm™ release
- **Labels:** the labels of the Helm™ release

4.2.3 OCI™ Distribution Specification API

4.2.3.1 Introduction

This clause provides an overview over the OCI™ Distribution Specification [5] which is profiled to the requirements on the OS container image management service interface exposed by the CIR as specified in ETSI GS NFV-IFA 040 [2]. The OCI™ Distribution Specification is based on the earlier published Docker™ Registry HTTP API v2. The overview covers the high-level API structure as well as the generic concepts for the data model of the managed resource objects.

4.2.3.2 API structure

The OCI™ Distribution Specification [5] is the protocol to facilitate the management of OS container images which are stored in a CIR. The API endpoints are prefixed by the API version and the repository name:

```
/{VERSION}/{repository_name}/
```

Additional information on the utilization of the repository name is provided in clause B.2.1 of the present document.

The OCI™ Distribution Specification [5] supports read and write operations on the CIR resource objects via the OCI™ Distribution Specification API endpoints. The OCI™ Distribution Specification differentiates the following resource objects managed via their resource endpoints:

- **tag:** a custom identifier of an OS container image
- **manifest:** a JSON document which specifies an artefact of an OS container image
- **blob:** the binary form of content that is stored by a registry, addressable by a digest

4.2.3.3 Data model concepts

The artefacts to be managed via the OCI™ Distribution Specification [5] are components of an OCI™ image, which are specified in the OCI™ Image Format Specification [6]. An OCI™ image consists of the following components:

- **Image Manifest:** a document describing the components that make up a OS container image
- **Image Index:** an annotated index of image manifests
- **Filesystem Layer:** a changeset that describes a OS container's filesystem
- **Image Configuration:** a document determining layer ordering and configuration of the OS container image suitable for translation into a OS container runtime bundle

5 NFV object model mapping to profiled solution objects

5.1 Managed Container Infrastructure Objects

5.1.1 Compute MCIOs

Selected Kubernetes® resource objects of the Workloads category are identified to map to the Compute MCIO type of the NFV object model, see clauses 5.2.1 and 6.4 in ETSI GS NFV-IFA 040 [2]. Table 5.1.1-1 lists the Kubernetes® resource objects which are mapped to the NFV objects of the Compute MCIO type.

Table 5.1.1-1: Kubernetes® resource objects mapped to NFV objects of Compute MCIO type

Kubernetes® resource object kind	Kubernetes® resource URI	Kubernetes® resource object description
CronJob	/apis/batch/v1/namespaces/{namespace}/cronjobs	Represents a recurring task that runs to completion and then stops.
DaemonSet	/apis/apps/v1/namespaces/{namespace}/daemonsets	Defines a set of Pods that provide local facilities of CIS cluster nodes. See note.
Deployment	/apis/apps/v1/namespaces/{namespace}/deployments	Represents a stateless application workload.
Job	/apis/batch/v1/namespaces/{namespace}/jobs	Represents a one-off task that runs to completion and then stops.
Pod	/api/v1/namespaces/{namespace}/pods	Represents the smallest deployable unit of an application workload as a group of one or more OS containers.
ReplicaSet	/apis/apps/v1/namespaces/{namespace}/replicasets	Represents a stable set of stateless application workloads.
StatefulSet	/apis/apps/v1/namespaces/{namespace}/statefulsets	Represents a stateful application workload.
NOTE: DaemonSet is considered as Compute MCIO type in case it provides resources in the context and as a part of an application.		

NOTE: The Kubernetes® "Container" resource object is not identified and mapped as individual Compute MCIO because it is always managed within the context of a Kubernetes® "Pod" resource object.

5.1.2 Storage MCIOs

Selected Kubernetes® resource objects of the Storage category are identified to map to the Storage MCIO type of the NFV object model, see clauses 5.2.1 and 6.5 in ETSI GS NFV-IFA 040 [2]. Table 5.1.2-1 lists the Kubernetes® resource objects which are mapped to the NFV objects of the Storage MCIO type.