ETSI GS NFV-SOL 015 V1.1.1 (2020-01)

Network Functions Virtualisation (NFV); Protocols and Data Models; Specification of Patterns and Conventions for RESTful NFV-MANO APIs

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
DGS/NFV-SOL015

Keywords
API, DATA, MANO, model, NFV, protocol

ETSI

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

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

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

Modal verbs terminology

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

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

1 Scope

The present document defines patterns and conventions for RESTful NFV-MANO API specifications, gives recommendations on API versioning and provides an API specification template.

The present document defines provisions to be followed by the ETSI NFV Industry Specification Group (ISG) when creating RESTful NFV-MANO API specifications. The provisions do not apply to implementations.

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-SOL 013: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

[2] IETF RFC 5789: "PATCH Method for HTTP".

NOTE: Available from https://tools.ietf.org/html/rfc5789.

[3] IETF RFC 7396: "JSON Merge Patch".

NOTE: Available from https://tools.ietf.org/html/rfc7396.

[4] IETF RFC 7232: "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

NOTE: Available from https://tools.ietf.org/html/rfc7232.

[5] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

NOTE: Available from https://tools.ietf.org/html/rfc3986.

[6] IETF RFC 7233: "Hypertext Transfer Protocol (HTTP/1.1): Range Requests".

NOTE: Available from https://tools.ietf.org/html/rfc7233.

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

[i.2] PlantUML website.

NOTE: Available from http://plantuml.com/.

[i.3] PlantUML tool download.

NOTE: Available from https://sourceforge.net/projects/plantuml/files/plantuml.jar/download.

[i.4] PlantUML reference guide.

NOTE: Available from http://plantuml.com/PlantUML_Language_Reference_Guide.pdf.

[i.5] ETSI NFV repository of OpenAPI™ files.

NOTE 1: Available from https://forge.etsi.org/rep/nfv/.

NOTE 2: OpenAPI Specification and OpenAPI Initiative and their respective logos, are trademarks of the Linux Foundation.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.1] apply.

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
BWC	BackWard Compatible
CR	Change Request
CRUD	Create, Read, Update, Delete
CTC	Change Type Code
ETag	Entity Tag
ETSI	European Telecommunications Standards Institute
GS	Group Specification
HATEOAS	Hypermedia As The Engine Of Application State
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IFA	InterFaces and Architecture
ISG	Industry Specification Group
JSON	JavaScript Object Notation
MANO	MANagement and Orchestration
NBWC	Non-BackWard Compatible
NFV	Network Functions Virtualisation
REST	REpresentational State Transfer
RFC	Request For Comments
SOL	SOLutions
URI	Uniform Resource Identifier

4 Conventions for names, strings and URIs

4.1 Case conventions

The following case conventions for names and strings are available for potential use in the RESTful NFV-MANO API specifications.

1) ALLUPPER

All letters of a string shall be uppercase letters. Digits may be used except at the first position. Other characters shall not be used.

EXAMPLES 1 and 2:

a) MANAGEMENTINTERFACE

b) ETSINFVMANAGEMENT2

2) alllower

All letters of a string shall be lowercase letters. Digits may be used except at the first position. Other characters shall not be used.

EXAMPLES 3 and 4:

a) managementinterface

b) etsinfvmanagement2

3) UPPER_WITH_UNDERSCORE

All letters of a string shall be uppercase letters. Digits may be used except at the first position. Word boundaries shall be represented by the underscore "_" character. Other characters shall not be used.

EXAMPLES 5 and 6:

a) MANAGEMENT_INTERFACE

b) ETSI_NFV_MANAGEMENT_2

4) lower_with_underscore

All letters of a string shall be lowercase letters. Digits may be used except at the first position. Word boundaries shall be represented by the underscore "_" character. Other characters shall not be used.

EXAMPLES 7 and 8:

a) management_interface

b) etsi_nfv_management_2

5) UpperCamel

The string shall be formed by concatenating words as follows: Each word shall start with an uppercase letter (this implies that the string starts with an uppercase letter). All other letters shall be lowercase letters. Digits may be used except at the first position. Other characters shall not be used. Words that are abbreviations shall follow the same scheme (i.e. first letter uppercase, all other letters lowercase).

EXAMPLES 9 and 10:

a) ManagementInterface

b) EtsiNfvManagement2

6) lowerCamel

The string shall follow the provisions defined for UpperCamel with the following difference: The first letter shall be lowercase (i.e. the first word starts with a lowercase letter).

EXAMPLES 11 and 12:

a) managementInterface

b) etsiNfvManagement2

4.2 Conventions for URI parts

Based on IETF RFC 3986 [5], the parts of the URI syntax that are relevant in the context of the RESTful NFV-MANO API specifications are as follows:

- *Path*, consisting of *segments*, separated by "/" (e.g. segment1/segment2/segment3)
- *Query*, consisting of pairs of parameter name and value (e.g. ?org=nfv&group=sol)

1) Path segment naming

a) The path segments of a resource URI which represent a constant string shall use the "lower_with_underscore" convention.

EXAMPLE 1: vnf_instances

b) If a resource represents a collection of entities, the last path segment of that resource's URI shall be a word in plural.

EXAMPLE 2: …/prefix/API/1_0/users

c) For resources that are not task resources, the last path segment of the resource URI should be a (composite) noun.

EXAMPLE 3: …/prefix/API/1_0/users

d) For resources that are task resources, the last path segment of the resource URI should be a verb, or start with a verb.

EXAMPLES 4 and 5:

- …/vnf_instances/{vnfInstanceId}/scale
- …/vnf_instances/{vnfInstanceId}/scale_to_level

e) A variable name which represents a single path segment or a sequence of one or more path segments of a resource URI shall use the "lowerCamel" convention and shall be surrounded by curly brackets. A variable in the path part of a resource URI represents a single path segment unless it is explicitly specified that it represents zero, one or more path segments.

EXAMPLE 6: {vnfInstanceId}

f) Once a variable is replaced at runtime by an actual string, the string shall follow the rules for a single path segment or one or more path segments defined in IETF RFC 3986 [5]. IETF RFC 3986 [5] disallows certain characters from use in a path segment. Each actual RESTful NFV-MANO API specification shall define this restriction to be followed when generating values for variables that represent a single path segment or one or more path segments, or propose a suitable encoding (such as percent-encoding according to IETF RFC 3986 [5]), to escape such characters if they can appear in input strings intended to be substituted for a path segment variable.

2) Query naming

a) Parameter names in queries shall use the "lower_with_underscore" convention.

EXAMPLE 7: ?working_group=SOL

b) Variables that represent actual parameter values in queries shall use the "lowerCamel" convention and shall be surrounded by curly brackets.

EXAMPLE 8: ?working_group={chooseAWorkingGroup}

c) Once a variable is replaced at runtime by an actual string, the convention defined in 1.f. shall apply to that string.

4.3 Conventions for names in data structures

The following syntax conventions shall be obeyed when defining the names for attributes and parameters in the RESTful NFV-MANO API data structures.

a) Names of attributes/parameters shall be represented using the "lowerCamel" convention.

EXAMPLE 1: vnfName

b) Names of arrays (i.e. those with cardinality 1..N or 0..N) shall be plural rather than singular.

EXAMPLES 2 and 3: users, extVirtualLinks

c) Identifiers of information elements defined in the corresponding "interface and information model" design stage specification (typically, ETSI NFV-IFA specification) using the name syntax "xyzStructureId" shall be represented using the name "id."

NOTE: In the aforementioned specifications, the identifiers in scope of this convention are attributes with names using the syntax "xyzStructureId" which are embedded in an information element named "XyzStructure" for the purpose of identifying and/or externally referencing an instance of that information element.

d) Each value of an enumeration types shall be represented using the "UPPER_WITH_UNDERSCORE" convention.

EXAMPLE 4: NOT_INSTATIATED

e) The names of data types shall be represented using the "UpperCamel" convention.

EXAMPLES 5 and 6: ResourceHandle, VnfInstance

5 Patterns

5.1 Pattern: Creating a resource (POST)

5.1.1 Description

This clause describes the "resource creation by POST" pattern, where the API consumer requests the API producer to create a new resource under a parent resource, which means that the URI identifying the created resource is under control of the API producer. This pattern shall be used for resource creation if the resource identifiers under the parent resource are managed by the API producer (see clause 5.2 for an alternative).

New resources are created on the origin server (API producer) as children of a parent resource. In order to request resource creation, the API consumer sends a POST request to the parent resource and includes a representation of the resource to be created. The API producer generates an identifier for the new resource that is unique for all child resources in the scope of the parent resource, and concatenates this with the resource URI of the parent resource to form the resource URI of the child resource. The API producer creates the new resource, and returns in a "201 Created" response a representation of the created resource along with a "Location" HTTP header that contains the resource URI of this resource.

Figure 5.1.1-1 illustrates creating a resource using POST.

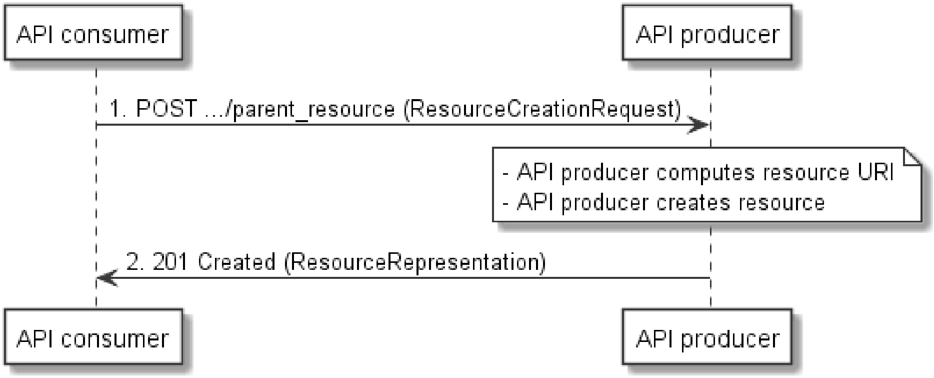

Figure 5.1.1-1: Resource creation flow by POST

5.1.2 Resource definition(s) and HTTP methods

The following resources are involved:

1) Parent resource: A container that can hold zero or more child resources.

2) Created resource: A child resource of a container resource that is created as part of the operation. The resource URI of the child resource is a concatenation of the resource URI of the parent resource with a string that is chosen by the API producer, and that is unique in the scope of the parent resource URI.

The HTTP method shall be POST.

5.1.3 Resource representation(s)

The entity body of the request shall contain a representation of the resource to be created. The entity body of the response shall contain a representation of the created resource.

NOTE: Compared to the entity body passed in the request, the entity body in the response may be different, as the resource creation process may have modified the information that has been passed as input, or generated additional attributes.

5.1.4 HTTP Headers

On success, the "Location" HTTP header shall be returned, and shall contain the URI of the newly created resource.

5.1.5 Response codes and error handling

On success, "201 Created" shall be returned. On failure, the appropriate error code shall be returned.

Resource creation can also be asynchronous in which case "202 Accepted" shall be returned instead of "201 Created". See clauses 5.11 and 5.12 for more details about asynchronous operations.

5.2 Pattern: Creating a resource (PUT)

5.2.1 Description

This clause describes the "resource creation by PUT" pattern, where the API consumer requests the API producer to create a new resource by providing the resource URI under which it expects the resource to be created, which means that the URI identifying the created resource is under control of the API consumer.

NOTE: The parent resource in this mode is implicit, i.e. it can be derived from the resource URI of the resource to be created by omitting the last path segment but is not provided explicitly in the request.

This pattern shall be used for resource creation if the resource identifiers under the parent resource are managed by the API consumer (see clause 5.1 for an alternative where the resource URI space is managed by the API producer). Typically, that alternative is safer as the API producer can prevent collisions. However, there are also valid use cases for creation by PUT, for instance when an API consumer manages different versions of a resource which are kept inside a "store" container, and the set of "store" containers is managed by the API producer i.e. "store" containers are created by POST.

In order to request resource creation, the API consumer sends a PUT request specifying the resource URI of the resource to be created and includes a representation of the resource to be created. The origin server (API producer) creates the new resource and returns in a "201 Created" response a representation of the created resource along with a "Location" HTTP header field that contains the resource URI of this resource.

Figure 5.2-1 illustrates creating a resource by PUT.

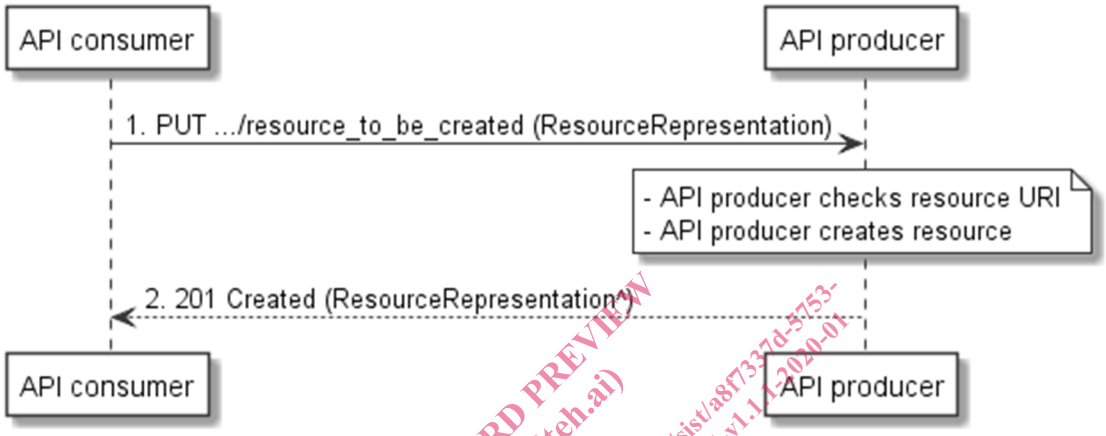

Figure 5.2-1: Resource creation by PUT

5.2.2 Resource definition(s) and HTTP methods

The following resource is involved:

1) Created resource: A resource that is created as part of the operation. The resource URI of that resource is passed by the API consumer in the PUT request.

The HTTP method shall be PUT.

5.2.3 Resource representation(s)

The entity body of the request shall contain a representation of the resource to be created. The entity body of the response shall contain a representation of the created resource.

NOTE: Compared to the entity body passed in the request (ResourceRepresentation in figure 5.2-1), the entitybody in the response (ResourceRepresentation^ in figure 5.2-1) may be different, as the resource creation process may have modified the information that has been passed as input.

5.2.4 HTTP Headers

On success, the "Location" HTTP header shall be returned, and shall contain the URI of the newly created resource.

5.2.5 Response codes and error handling

The API producer shall check whether the resource URI of the resource to be created does not conflict with the resource URIs of existing resources (i.e. whether or not the resource requested to be created already exists).

In case the resource does not yet exist:

- Upon successful resource creation, "201 Created" shall be returned. Upon failure, the appropriate error code shall be returned.
- Resource creation can also be asynchronous in which case "202 Accepted" shall be returned instead of "201 Created". See clauses 5.11 and 5.12 for more details about asynchronous operations.

In case the resource already exists:

- If the "Update by PUT" operation is not supported for the resource, the request shall be rejected with "403 Forbidden", and a "ProblemDetails" payload should be included to provide more information about the error.
- If the "Update by PUT" operation is supported for the resource, interpret the request as an update request, i.e. the request shall be processed as defined in clause 5.6.

5.3 Pattern: Reading a resource (GET)

5.3.1 Description

This pattern obtains a representation of the resource, i.e. reads a resource, by using the HTTP GET method. For most resources, the GET method should be supported. An exception is task resources (see clause 5.8); these cannot be read.

See clause 5.3 for the related "query" pattern which allows to obtain a representation of a resource that has child resources and allows to influence the content of the representation using query parameters.

Figure 5.3.1-1 illustrates reading a resource.

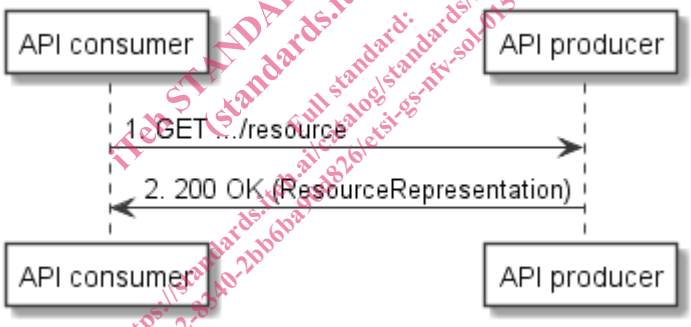

Figure 5.3.1-1: Flow of reading a resource

5.3.2 Resource definition(s) and HTTP methods

This pattern is applicable to any resource that can be read. The HTTP method shall be GET.

5.3.3 Resource representation(s)

The entity body of the request shall be empty; the entity body of the response shall contain a representation of the resource that was read, if successful.

5.3.4 HTTP Headers

There are no specific provisions for HTTP headers for this pattern.

5.3.5 Response codes and error handling

On success, "200 OK" shall be returned. On failure, the appropriate error code shall be returned.