ETSI GS NFV-SOL 013 V2.6.1 (2019-03)

Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Specification of common aspects 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-SOL013ed261

Keywords

API, 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

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

Foreword

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

Modal verbs terminology

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

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

1 Scope

The present document specifies common aspects of RESTful protocols and data models for ETSI NFV management and orchestration (MANO) interfaces.

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] IETF RFC 2818: "HTTP Over TLS".

NOTE: Available at https://tools.ietf.org/html/rfc2818.

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

NOTE: Available at https://tools.ietf.org/html/rfc3339.

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

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

[4] IETF RFC 4918: "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)".

NOTE: Available at https://tools.ietf.org/html/rfc4918.

[5] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

NOTE: Available at https://tools.ietf.org/html/rfc5246.

[6] IETF RFC 6585: "Additional HTTP Status Codes".

NOTE: Available at https://tools.ietf.org/html/rfc6585.

[7] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

NOTE: Available from https://tools.ietf.org/html/rfc6749.

[8] IETF RFC 6750: "The OAuth 2.0 Authorization Framework: Bearer Token Usage".

NOTE: Available from https://tools.ietf.org/html/rfc6750.

[9] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at https://tools.ietf.org/html/rfc8259.

[10] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".

NOTE: Available at https://tools.ietf.org/html/rfc7231.

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

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

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

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

[13] IETF RFC 7235: "Hypertext Transfer Protocol (HTTP/1.1): Authentication".

NOTE: Available at https://tools.ietf.org/html/rfc7235.

[14] IETF RFC 7617: "The 'Basic' HTTP Authentication Scheme".

NOTE: Available from https://tools.ietf.org/html/rfc7617.

[15] IETF RFC 7807: "Problem Details for HTTP APIs".

NOTE: Available at https://tools.ietf.org/html/rfc7807.

[16] IETF RFC 6901: "JavaScript Object Notation (JSON) Pointer".

NOTE: Available at https://tools.ietf.org/html/rfc6901.

[17] IETF RFC 8288: "Web Linking".

NOTE: Available at https://tools.ietf.org/html/rfc8288.

[18] Semantic Versioning 2.0.0.

NOTE: Available at https://semver.org/.

[19] IETF RFC 4229: "HTTP Header Field Registrations".

NOTE: Available at https://tools.ietf.org/html/rfc4229.

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] ETSI TS 133 310: "Universal Mobile Telecommunications System (UMTS); LTE; Network Domain Security (NDS); Authentication Framework (AF) (3GPP TS 33.310)".

[i.3] Hypertext Transfer Protocol (HTTP) Status Code Registry at IANA.

NOTE: Available at http://www.iana.org/assignments/http-status-codes.

[i.4] ETSI NFV OpenAPI repository.

NOTE: Available at https://forge.etsi.org/rep/nfv/.

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
EM	Element Manager
ETSI	European Telecommunications Standards Institute
GMT	Greenwich Mean Time
GS	Group Specification
HATEOAS	Hypermedia As The Engine Of Application State
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
MAC	Medium Access Control
MANO	Management and Orchestration
MIME	Multipurpose Internet Mail Extensions
NFV	Network Functions Virtualisation
NFVO	NFV Orchestrator
REST	Representational State Transfer
RFC	Request For Comments
TLS	Transport Layer Security
URI	Uniform Resource Identifier
VIM	Virtualised Infrastructure Manager
VNF	Virtualised Network Function
VNFM	VNF Manager

4 HTTP usage

4.1 URI structure and supported content formats

This clause specifies the URI prefix and the supported formats applicable to the APIs defined in the present document.

All resource URIs of the APIs shall have the following prefix, except the "API versions" resource which shall follow the rules specified in clause 9.3:

{apiRoot}/{apiName}/{apiMajorVersion}/

where:

{apiRoot}	indicates the scheme ("http" or "https"), the host name and optional port, and an optional sequence of path segments that together represent a prefix path.

EXAMPLE: http://orchestrator.example.com/nfv_apis/abc

{apiName} indicates the interface name in an abbreviated form. The {apiName} of each interface is defined in the clause specifying the corresponding interface.

{apiMajorVersion} indicates the current major version (see clause 9.1) of the API and is defined in the clause specifying the corresponding interface.

For HTTP requests and responses that have a body, the content format JSON (see IETF RFC 8259 [9]) shall be supported. The JSON format shall be signalled by the content type "application/json".

All APIs shall support and use HTTP over TLS (also known as HTTPS) (see IETF RFC 2818 [1]). TLS version 1.2 as defined by IETF RFC 5246 [5] shall be supported.

NOTE 1: The HTTP protocol elements mentioned in the present document originate from the HTTP specification; HTTPS runs the HTTP protocol in a TLS layer. The present document therefore uses the statement above to mention "HTTP request", "HTTP header", etc., without explicitly calling out whether or not these are run over TLS.

NOTE 2: There are a number of best practices and guidelines how to configure and implement TLS 1.2 in a secure manner, as security threats evolve. A detailed specification of those is beyond the scope of the present document; the reader is referred to external documentation such as annex E of ETSI TS 133 310 [i.2].

All resource URIs of the API shall comply with the URI syntax as defined in IETF RFC 3986 [3]. An implementation that dynamically generates resource URI parts (individual path segments, sequences of path segments that are separated by "/", query parameter values) shall ensure that these parts only use the character set that is allowed by IETF RFC 3986 [3] for these parts.

NOTE 3: This means that characters not part of this allowed set are escaped using percent-encoding as defined by IETF RFC 3986 [3].

Unless otherwise specified explicitly, all request URI parameters that are part of the path of the resource URI shall be individual path segments, i.e. shall not contain the "/" character.

NOTE 4: A request URI parameter is denoted by a string in curly brackets, e.g. {subscriptionId}.

4.2 Usage of HTTP header fields

4.2.1 Introduction

HTTP headers are components of the header section of the HTTP request and response messages. They contain the information about the server/client and metadata of the transaction. The use of HTTP header fields shall comply with the provisions defined for those header fields in the specifications referenced from tables 4.2.2-1 and 4.2.3-1. The following clauses describe the HTTP header fields that are explicitly mentioned in the present document.

4.2.2 Request header fields

This clause describes the usage of HTTP header fields of the request messages applicable to the APIs defined in the present document. The HTTP header fields used in the request messages are specified in table 4.2.2-1.

Table 4.2.2-1: Header fields supported in the request message

Header field name	Reference	Example	Descriptions
Accept	IETF RFC 7231 [10]	application/json	Content-Types that are acceptable for the response. This header field shall be present if the response is expected to have a non-empty message body.
Content-Type	IETF RFC 7231 [10]	application/json	The MIME type of the body of the request. This header field shall be present if the request has a non-empty message body.
Authorization	IETF RFC 7235 [13]	Bearer mF_9.B5f-4.1JqM	The authorization token for the request. Details are specified in clause 8.3.
Range	IETF RFC 7233 [12]	1 000-2 000	Requested range of bytes from a file.
Version	IETF RFC 4229 [19]	1.2.0 or 1.2.0-impl:example.com:myProduct:4	Version of the API requested to use when responding to this request.

4.2.3 Response header fields

This clause describes the usage of HTTP header fields of the response messages applicable to the APIs defined in the present document. The HTTP header fields used in the response messages are specified in table 4.2.3-1.

Table 4.2.3-1: Header fields supported in the response message

Header field name	Reference	Example	Descriptions
Content-Type	IETF RFC 7231 [10]	application/json	The MIME type of the body of the response. This header field shall be present if the response has a non-empty message body.
Location	IETF RFC 7231 [10]	http://www.example.com/vnflcm/v1/vnf_instances/123	Used in redirection, or when a new resource has been created. This header field shall be present if the response status code is 201 or 3xx. In the present document this header field is also used if the response status code is 202 and a new resource was created.
WWW-Authenticate	IETF RFC 7235 [13]	Bearer realm="example"	Challenge if the corresponding HTTP request has not provided authorization, or error details if the corresponding HTTP request has provided an invalid authorization token.
Accept-Ranges	IETF RFC 7233 [12]	bytes	Used by the server to signal whether or not it supports ranges for certain resources.
Content-Range	IETF RFC 7233 [12]	bytes 21 010 - 47 021/47 022	Signals the byte range that is contained in the response, and the total length of the file.
Retry-After	IETF RFC 7231 [10]	Fri, 31 Dec 1999 23:59:59 GMT or 120	Used to indicate how long the user agent ought to wait before making a follow-up request. It can be used with 503 responses. The value of this field can be an HTTP-date or a number of seconds to delay after the response is received.

Header field name	Reference	Example	Descriptions
Link	IETF RFC 8288 [17]	<http://example.com/resources?nextpage_opaque_marker=abc123>; rel="next"	Reference to other resources. Used for paging in the present document, see clause 5.4.2.1.
Version	IETF RFC 4229 [19]	1.2.0 or 1.2.0-impl:example.com:myProduct:4	Version of the API requested to use when responding to this request.

5 Result set control

5.1 Introduction

This clause specifies procedures that allow to control the size of the result set of GET requests w.r.t. the number of entries in a response list (using attribute-based filtering) or w.r.t. the number of attributes returned in a response (using attribute selection).

5.2 Attribute-based filtering

5.2.1 Overview and example (informative)

Attribute-based filtering allows to reduce the number of objects returned by a query operation. Typically, attribute-based filtering is applied to a GET request that reads a resource which represents a list of objects (e.g. child resources). Only those objects that match the filter are returned as part of the resource representation in the payload body of the GET response.

Attribute-based filtering can test a simple (scalar) attribute of the resource representation against a constant value, for instance for equality, inequality, greater or smaller than, etc. Attribute-based filtering is requested by adding a set of URI query parameters, the "attribute-based filtering parameters" or "filter" for short, to a resource URI.

The following example illustrates the principle. Assume a resource "container" with the following objects:

EXAMPLE 1: Objects:

```
obj1: {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]}
obj2: {"id":456, "weight":500, "parts":[{"id":3, "color":"green"}, {"id":4, "color":"blue"}]}
```

A GET request on the "container" resource would deliver the following response:

EXAMPLE 2: Unfiltered GET:

```
Request:
GET …/container
Response:
[
    {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]},
    {"id":456, "weight":500, "parts":[{"id":3, "color":"green"}, {"id":4, "color":"blue"}]}
]
```

A GET request with a filter on the "container" resource would deliver the following response:

EXAMPLE 3: GET with filter:

```
Request:
GET …/container?filter=(eq,weight,100)
Response:
[
    {"id":123, "weight":100, "parts":[{"id":1, "color":"red"}, {"id":2, "color":"green"}]}

]
```

For hierarchically-structured data, filters can also be applied to attributes deeper in the hierarchy. In case of arrays, a filter matches if any of the elements of the array matches. In other words, when applying the filter "(eq,parts/color,green)" to the objects in Example 1, the filter matches obj1 when evaluating the second entry in the "parts" array of obj1 and matches obj2 already when evaluating the first entry in the "parts" array of obj2. As the result, both obj1 and obj2 match the filter.

If a filter contains multiple sub-parts that only differ in the leaf attribute (i.e. they share the same attribute prefix), they are evaluated together per array entry when traversing an array. As an example, the two expressions in the filter "(eq,parts/color,green);(eq,parts/id,3)" would be evaluated together for each entry in the array "parts". As the result, obj2 matches the filter.

5.2.2 Specification

An attribute-based filter shall be represented by a URI query parameter named "filter". The value of this parameter shall consist of one or more strings formatted according to "simpleFilterExpr", concatenated using the ";" character:

```
simpleFilterExprOne      := <opOne>","<attrName>["/"<attrName>]*","<value>
simpleFilterExprMulti    := <opMulti>","<attrName>["/"<attrName>]*","<value>[","<value>]*
simpleFilterExpr         := "("<simpleFilterExprOne>")" | "("<simpleFilterExprMulti>")"
filterExpr               := <simpleFilterExpr>[";"<simpleFilterExpr>]*
filter                   := "filter"=<filterExpr>
opOne                    := "eq" | "neq" | "gt" | "lt" | "gte" | "lte"
opMulti                  := "in" | "nin" | "cont" | "ncont"
attrName                 := string
value                    := string
```

where:

```
*    zero or more occurrences
[]   grouping of expressions to be used with *
""   quotation marks for marking string constants
<>   name separator
|    separator of alternatives
```


"AttrName" is the name of one attribute in the data type that defines the representation of the resource. The slash ("/") character in "simpleFilterExprOne" and "simpleFilterExprMulti" allows concatenation of <attrName> entries to filter by attributes deeper in the hierarchy of a structured document. The elements "opOne" and "opMulti" stand for the comparison operators (accepting one comparison value or a list of such values). If the expression has concatenated <attrName> entries, it means that the operator is applied to the attribute addressed by the last <attrName> entry included in the concatenation. All simple filter expressions are combined by the "AND" logical operator, denoted by ";".

In a concatenation of <attrName> entries in a <simpleFilterExprOne> or <simpleFilterExprMulti>, the rightmost <attrName> entry is called "leaf attribute". The concatenation of all "attrName" entries except the leaf attribute is called the "attribute prefix". If an attribute referenced in an expression is an array, an object that contains a corresponding array shall be considered to match the expression if any of the elements in the array matches all expressions that have the same attribute prefix.

The leaf attribute of a <simpleFilterExprOne> or <simpleFilterExprMulti> shall not be structured but shall be of a simple (scalar) type such as String, Number, Boolean or DateTime, or shall be an array of simple (scalar) values. Attempting to apply a filter with a structured leaf attribute shall be rejected with "400 Bad request". A <filterExpr> shall not contain any invalid <simpleFilterExpr> entry.

The operators listed in table 5.2.2-1 shall be supported.