ETSI GS NFV-SOL 002 V2.4.1 (2018-02)

GROUP SPECIFICATION

Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Ve-Vnfm Reference Point

Disclaimer

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 only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

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

ETSI

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 a set of RESTful protocols and data models fulfilling the requirements specified in ETSI GS NFV-IFA 008 [1] for the interfaces used over the Ve-Vnfm reference point.

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 http://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 008: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification".

[2]   IETF RFC 2818: "HTTP Over TLS".

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

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

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

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

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

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

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

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

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

[7]   IETF RFC 5646: "Tags for Identifying Languages".

NOTE: Available at https://tools.ietf.org/html/rfc5646.

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

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

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

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

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

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

[11]   IETF RFC 7159: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at https://tools.ietf.org/html/rfc7159.

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

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

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

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

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

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

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

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

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

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

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

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

[18]   Recommendation ITU-T X.733: "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

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

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

[i.4]   Hypertext Transfer Protocol (HTTP) Status Code Registry at IANA.

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

[i.5]   ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

3 Definitions and abbreviations

3.1 Definitions

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

LCM workflow: set of operations, including resource management operations towards the VIM, that are executed by the VNFM to perform a lifecycle management operation

NOTE: Examples for LCM workflows are VNFM-internal procedures associated with an LCM operation, and LCM scripts contained in the VNF package.

3.2 Abbreviations

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

API	Application Programming Interface
CP	Connection Point
CPD	CP Descriptor
DHCP	Dynamic Host Configuration Protocol
EM	Element Manager
ETSI	European Telecommunications Standards Institute
FM	Fault Management
GMT	Greenwich Mean Time
GS	Group Specification
GUI	Graphical User Interface
HATEOAS	Hypermedia As The Engine Of Application State
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IETF	Internet Engineering Task Force
IFA	Interfaces and Architecture
IP	Internet Protocol
ITU-T	International Telecommunications Union - Telecommunication
JSON	JavaScript Object Notation
LCCN	Life Cycle Change Notification
LCM	Lifecycle Management
MAC	Media Access Control
MANO	Management and Orchestration
MIME	Multipurpose Internet Mail Extensions
NFV	Network Functions Virtualisation
NFVO	NFV Orchestrator
NS	Network Service
PM	Performance Management
REST	Representational State Transfer
RFC	Request for Comments
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
URI	Uniform Resource Identifier
VDU	Virtualisation Deployment Unit
VIM	Virtualised Infrastructure Manager
VL	Virtual Link
VLD	VL Descriptor
VNF	Virtualised Network Function
VNFC	VNF Component
VNFD	VNF Descriptor
VNFM	VNF Manager
VNFLCM	Virtualised Network Function Lifecycle Management

4   General aspects

4.1   Overview

The present document defines the protocol and data model for the following interfaces used over the Ve-Vnfm reference point, in the form of RESTful Application Programming Interfaces (APIs) specifications:

- VNF Lifecycle Management interface (as produced by the VNFM towards the EM/VNF)

- VNF Performance Management interface (as produced by the VNFM towards the EM)

- VNF Fault Management interface (as produced by the VNFM towards the EM)

- VNF Indicator interface (as produced by the EM/VNF towards the VNFM)

- VNF Configuration interface (as produced by the VNF towards the VNFM)

The design of the protocol and data model for the above interfaces is based on the information model and requirements defined in ETSI GS NFV-IFA 008 [1]. In clause 4, general aspects such as URI structure and supported content formats, general procedures, common data types and authorization of API requests and notifications are specified.

In the subsequent clauses, the protocol and data models for the individual interfaces are specified. Per interface, the resource structure with associated HTTP methods is defined and applicable flows are provided. Further, the resources and the data model are specified in detail.

Annex A provides the mapping of the combination of resources and methods defined in the present document to the operations defined in ETSI GS NFV-IFA 008 [1]. Annex B contains explanations of key concepts.

Even though the different interfaces defined in the present document are related, implementations shall not assume a particular order of messages that arrive via different interfaces.

4.2   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:

{apiRoot}/{apiName}/{apiVersion}/

where:

{apiRoot}	indicates the scheme ("http" or "https"), the host name and optional port, and an optional prefix path.
{apiName}	indicates the interface name in an abbreviated form. The {apiName} of each interface is defined in the clause specifying the corresponding interface.
{apiVersion}	indicates the current version 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 7159 [11]) 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 [2]). TLS version 1.2 as defined by IETF RFC 5246 [6] 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.3].

All resource URIs of the API shall comply with the URI syntax as defined in IETF RFC 3986 [4]. An implementation
that dynamically generates resource URI parts (path segments, query parameter values) shall ensure that these parts
only use the character set that is allowed by IETF RFC 3986 [4] 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 [4].

4.3   Common procedures

4.3.1   Introduction

This clause specifies procedures applicable to all interfaces.

4.3.2   Attribute-based filtering

4.3.2.1   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:2, color:green}]}
```

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:2, 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?weight.eq=100 or GET …/container?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 "parts.color.eq=green" (or "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 filter "parts.color.eq=green"&"parts.id.eq=3" (or "parts.color=green&parts.id=3") would be evaluated together for each entry in the array "parts". As the result, obj2 matches the filter.

4.3.2.2   Specification

A set of filter parameters shall be represented as a part of the URI query string. This means, it shall consist of one or more strings formatted according to "simpleFilterExpr", concatenated using the "&" character:

```
simpleFilterExpr := <attrName>["."<attrName>]*["."<op>]?"="<value>[','<value>]*
filterExpr       := <simpleFilterExpr>["&"<simpleFilterExpr>]*
op               := "eq" | "neq" | "gt" | "lt" | "gte" | "lte" | "cont" |
                    "ncont"
attrName         := string
value                := scalar value
where:
*   zero or more occurrences
?   zero or one occurrence
[]  grouping of expressions to be used with ? and *
""  quotation marks for marking string constants
<>  name separator
```

"AttrName" is the name of one attribute in the data type that defines the representation of the resource. The dot (".") character in "simpleFilterExpr" allows concatenation of <attrName> entries to filter by attributes deeper in the hierarchy of a structured document. "Op" stands for the comparison operator. If the expression has concatenated <attrName> entries, it means that the operator "op" 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.

In a concatenation of <attrName> entries in a <simpleFilterExpr>, 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 <simpleFilterExpr> shall not be structured, but shall be of a simple (scalar) type such as String, Number or DateTime, or shall be an array of simple (scalar) attributes. 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 "op" listed in table 4.3.2.2-1 shall be supported.

Table 4.3.2.2-1: Operators for attribute-based filtering

Operator <op>	Meaning
<attrName>.eq=<value>[,<value>]*	Attribute **equal** to one of the values in the list
<attrName>=<value>[,<value>]*	Alternative representation of equality. See note.
<attrName>.neq=<value>[,<value>]*	Attribute **not equal** to any of the values in the list
<attrName>.gt=<value>	Attribute **greater than** <value>
<attrName>.gte=<value>	Attribute **greater than or equal** to <value>
<attrName>.lt=<value>	Attribute **less than** <value>
<attrName>.lte=<value>	Attribute **less than or equal** to <value>
<attrName>.cont=<value>[,<value>]*	Attribute **contains** (at least) one of the values in the list
<attrName>.ncont=<value>[,<value>]*	Attribute **does not contain** any of the values in the list
NOTE: This representation shall not be used for attributes whose name is equal to the name of a defined URI query parameter.	

All objects that match the filter shall be returned as response to a GET request that contains a filter.

A <value> entry shall contain a scalar value, such as a number or string. The content of a <value> entry shall be formatted the same way as the representation of the related attribute in the resource representation, for instance as String, Integer, Boolean or DateTime. Attribute-based filters are supported for certain resources. Details are defined in the clauses specifying the actual resources.

4.3.3   Attribute selectors

4.3.3.1   Overview and example (informative)

Certain resource representations can become quite big, in particular, if the resource is a container for multiple sub-resources, or if the resource representation itself contains a deeply-nested structure. In these cases, it can be desired to reduce the amount of data exchanged over the interface and processed by the API consumer application. On the other hand, it can also be desirable that a "drill-deep" for selected parts of the omitted data can be initiated quickly.

An attribute selector allows the API consumer to choose which attributes it wants to be contained in the response. Only attributes that are not required to be present, i.e. those with a lower bound of zero on their cardinality (e.g. 0..1, 0..N) and that are not conditionally mandatory, are allowed to be omitted as part of the selection process. Attributes can be marked for inclusion or exclusion.

If an attribute is omitted, a link to a resource may be added where the information of that attribute can be fetched. Such approach is known as HATEOAS which is a common pattern in REST, and enables drilling down on selected issues without having to repeat a request that may create a potentially big response.

4.3.3.2   Specification

4.3.3.2.1   GET request

The URI query parameters for attribute selection are defined in table 4.3.3.2.1-1.

In the provisions below, "complex attributes" are assumed to be those attributes that are structured, or that are arrays.

Table 4.3.3.2.1-1: Attribute selector parameters

Parameter	Definition
all_fields	This URI query parameter requests that all complex attributes are included in the response, including those suppressed by exclude_default. It is inverse to the "exclude_default" parameter. The API producer shall support this parameter for certain resources. Details are defined in the clauses specifying the actual resources.
fields	This URI query parameter requests that only the listed complex attributes are included in the response. The parameter shall be formatted as a list of attribute names. An attribute name shall either be the name of an attribute, or a path consisting of the names of multiple attributes with parent-child relationship, separated by ".". Attribute names in the list shall be separated by comma (","). Valid attribute names for a particular GET request are the names of all complex attributes in the expected response that have a lower cardinality bound of 0 and that are not conditionally mandatory. The API producer should support this parameter for certain resources. Details are defined in the clauses specifying the actual resources.
exclude_fields	This URI query parameter requests that the listed complex attributes are excluded from the response. For the format, eligible attributes and support by the API producer, the provisions defined for the "fields" parameter shall apply.
exclude_default	Presence of this URI query parameter requests that a default set of complex attributes shall be excluded from the response. The default set is defined per resource in the present document. Not every resource will necessarily have such a default set. Only complex attributes with a lower cardinality bound of zero that are not conditionally mandatory can be included in the set. The API producer shall support this parameter for certain resources. Details are defined in the clauses specifying the actual resources. This parameter is a flag. i.e. it has no value. If a resource supports attribute selectors and none of the attribute selector parameters is specified in a GET request, the "exclude_default" parameter shall be assumed as the default.

4.3.3.2.2   GET response

Table 4.3.3.2.2-1 defines the valid parameter combinations.in a GET request and their effect on the GET response.

Table 4.3.3.2.2-1: Valid combinations of attribute selector parameters

Parameter combination	The GET response shall include…
(none)	… same as "exclude_default"
all_fields	… all attributes
fields=<list>	… all attributes except all complex attributes with minimum cardinality of zero that are not conditionally mandatory, and that are not provided in <list>
exclude_fields=<list>	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory, and that are provided in <list>
exclude_default	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory, and that are part of the "default exclude set" defined in the present specification for the particular resource
exclude_default and fields=<list>	… all attributes except those complex attributes with a minimum cardinality of zero that are not conditionally mandatory and that are part of the "default exclude set" defined in the present specification for the particular resource, but that are not part of <list>

If complex attributes were omitted in a GET response, the response may contain a number of links that allow to obtain directly the content of the omitted attributes. Such links shall be embedded into a structure named "_links" at the same level as the omitted attribute. That structure shall contain one entry for each link, named as the omitted attribute, and containing an "href" attribute that contains the URI of a resource that can be read with GET to obtain the content of the omitted attribute. A link shall not be present if the attribute is not present in the underlying resource representation. The resource URI structure of such links is not standardized, but may be chosen by the VNFM implementation. Performing a GET request on such a link shall return a representation that contains the content of the omitted attribute.

   EXAMPLE:

```
"_links" : {
"vnfcs" : {"href" : ".../vnflcm/v1/vnf_instances/1234/vnfcs"},
"extVirtualLinks" : {"href" : ".../vnflcm/v1/_dynamic/7d6bef4e-d86b-4abc-97ed-9dc9b951f206"}
}
```

4.3.4   Usage of HTTP header fields

4.3.4.1   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 4.3.4.2-1 and 4.3.4.3-1. The following clauses describe the HTTP header fields that are explicitly mentioned in the present document.

4.3.4.2   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 4.3.4.2-1.

Table 4.3.4.2-1: Header fields supported in the request message

Header field name	Reference	Example	Descriptions
Accept	IETF RFC 7231 [12]	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.

Header field name	Reference	Example	Descriptions
Content-Type	IETF RFC 7231 [12]	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 [14]	Bearer mF_9.B5f-4.1JqM	The authorization token for the request. Details are specified in clause 4.5.3.

4.3.4.3   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 4.3.4.3-1.

Table 4.3.4.3-1: Header fields supported in the response message

Header field name	Reference	Example	Descriptions
Content-Type	IETF RFC 7231 [12]	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 [12]	http://www.example.com/vnf lcm/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 [14]	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.
Retry-After	IETF RFC 7231 [12]	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.

4.3.5   Error reporting

4.3.5.1   Introduction

In RESTful interfaces, application errors are mapped to HTTP errors. Since HTTP error information is generally not enough to discover the root cause of the error, additional application specific error information is typically delivered. The following clauses define such a mechanism to be used by the interfaces specified in the present document.

4.3.5.2   General mechanism

When an error occurs that prevents the API producer from successfully fulfilling the request, the HTTP response shall include in the response a status code in the range 400..499 (client error) or 500.599 (server error) as defined by the HTTP specification (see IETF RFC 7231 [12], IETF RFC 7232 [13], and IETF RFC 7235 [14], as well as by IETF RFC 6585 [8]. In addition, the response body should contain a JSON representation of a "ProblemDetails" data structure according to IETF RFC 7807 [17] that provides additional details of the error. In that case, as defined by IETF RFC 7807 [17], the "Content-Type" HTTP header shall be set to "application/problem+json".

4.3.5.3   Type: ProblemDetails

The definition of the general "ProblemDetails" data structure from IETF RFC 7807 [17] is reproduced in table 4.3.5.3-1. Compared to the general framework defined in IETF RFC 7807 [17], the "status" and "detail" attributes are mandated to be included by the present specification, to ensure that the response contains additional textual information about an error. IETF RFC 7807 [17] foresees extensibility of the "ProblemDetails" type. It is possible that particular APIs in the present document, or particular implementations, define extensions to define additional attributes that provide more information about the error.

The description column only provides some explanation of the meaning to facilitate understanding of the design. For a full description, see IETF RFC 7807 [17].

Table 4.3.5.3-1: Definition of the ProblemDetails data type

Attribute name	Data type	Cardinality	Description
type	URI	0..1	A URI reference according to IETF RFC 3986 [4] that identifies the problem type. It is encouraged that the URI provides human-readable documentation for the problem (e.g. using HTML) when dereferenced. When this member is not present, its value is assumed to be "about:blank".
title	String	0..1	A short, human-readable summary of the problem type. It should not change from occurrence to occurrence of the problem, except for purposes of localization. If type is given and other than "about:blank", this attribute shall also be provided.
status	Integer	1	The HTTP status code for this occurrence of the problem.
detail	String	1	A human-readable explanation specific to this occurrence of the problem.
instance	URI	0..1	A URI reference that identifies the specific occurrence of the problem. It may yield further information if dereferenced.
(additional attributes)	Not specified.	0..N	Any number of additional attributes, as defined in a specification or by an implementation.

NOTE: It is expected that the minimum set of information returned in ProblemDetails consists of "status" and "detail". For the definition of specific "type" values as well as extension attributes by implementations, guidance can be found in IETF RFC 7807 [17].

4.3.5.4   Common error situations

The following common error situations are applicable on all REST resources and related HTTP methods specified in the present document, and shall be handled as defined in the present clause:

400 Bad Request: If the request is malformed or syntactically incorrect (e.g. if the request URI contains incorrect query parameters or a syntactically incorrect payload body), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and should include in the "detail" attribute more information about the source of the problem.

400 Bad Request: If the request contains a malformed access token, the API Producer should respond with this response. The details of the error shall be returned in the WWW-Authenticate HTTP header, as defined in IETF RFC 6750 [10] and IETF RFC 7235 [14]. The ProblemDetails structure may be provided.

400 Bad Request: If there is an application error related to the client's input that cannot be easily mapped to any other HTTP response code ("catch all error"), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and shall include in the "detail" attribute more information about the source of the problem.

NOTE 1: It is by design to represent this application error situation with the same HTTP error response code as the previous one.

401 Unauthorized: If the request contains no access token even though one is required, or if the request contains an authorization token that is invalid (e.g. expired or revoked), the API producer should respond with this response. The details of the error shall be returned in the WWW-Authenticate HTTP header, as defined in IETF RFC 6750 [10] and IETF RFC 7235 [14]. The ProblemDetails structure may be provided.

403 Forbidden: If the API consumer is not allowed to perform a particular request to a particular resource, the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided. It should include in the "detail" attribute information about the source of the problem, and may indicate how to solve it.

404 Not Found: If the API producer did not find a current representation for the resource addressed by the URI passed in the request, or is not willing to disclose that one exists, it shall respond with this response code. The "ProblemDetails" structure may be provided, including in the "detail" attribute information about the source of the problem, e.g. a wrong resource URI variable.

405 Method Not Allowed: If a particular HTTP method is not supported for a particular resource, the API producer shall respond with this response code. The "ProblemDetails" structure may be omitted in that case.

406 Not Acceptable: If the "Accept" HTTP header does not contain at least one name of a content type that is acceptable to the API producer, the API producer shall respond with this response code. The "ProblemDetails" structure may be omitted in that case.

422 Unprocessable Entity: If the payload body of a request contains syntactically correct data (e.g. well-formed JSON) but the data cannot be processed (e.g. because it fails validation against a schema), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and should include in the "detail" attribute more information about the source of the problem.

NOTE 2: This error response code is only applicable for methods that have a request body.

500 Internal Server Error: If there is an application error not related to the client's input that cannot be easily mapped to any other HTTP response code ("catch all error"), the API producer shall respond with this response code. The "ProblemDetails" structure shall be provided, and shall include in the "detail" attribute more information about the source of the problem.

503 Service Unavailable: If the API producer encounters an internal overload situation of itself or of a system it relies on, it should respond with this response code, following the provisions in IETF RFC 7231 [12] for the use of the "Retry-After" HTTP header and for the alternative to refuse the connection. The "ProblemDetails" structure may be omitted.

NOTE 3: The error handling defined above only applies to REST resources defined in the present document. For the token endpoint defined in IETF RFC 6749 [9] and re-used in the present document as defined in clause 4.5.3, the error handling provisions are defined in clause 4.5.3.

4.3.5.5   Overview of HTTP error status codes

Table 4.3.5.5-1 lists the HTTP error status codes that are explicitly mentioned in the present document. The full definition of each error code can be obtained from the referenced specification.

Table 4.3.5.5-1: HTTP error status codes used in the present document

Code	Status text	Reference	Explanation
400	Bad Request	IETF RFC 7231 [12] IETF RFC 6750 [10] IETF RFC 7235 [14]	Required information for the request was missing, the request had syntactical errors, or the request contains a malformed access token or malformed credentials. In the present document, this code is also used as "catch-all" code for client errors.
401	Unauthorized	IETF RFC 7235 [14]	Client is required to include valid credentials in the request. See clause 4.5.3.
403	Forbidden	IETF RFC 7231 [12]	The client is not allowed to perform the request on that resource.
404	Not Found	IETF RFC 7231 [12]	The requested URI was not found. A reason can e.g. be that resource URI variables were set wrongly.
405	Method Not Allowed	IETF RFC 7231 [12]	See clause 4.3.5.4.
406	Not Acceptable	IETF RFC 7231 [12]	See clause 4.3.5.4.

Code	Status text	Reference	Explanation
409	Conflict	IETF RFC 7231 [12]	Another request is in progress that prohibits the fulfillment of the current request, or the current resource state is inconsistent with the request.
412	Precondition Failed	IETF RFC 7232 [13]	This code is used in conjunction with conditional requests (typically used to protect resources consistency when using PUT or PATCH in a multi-client scenario) to indicate that a precondition has failed.
422	Unprocessable Entity	IETF RFC 4918 [5]	The server understands the content type of the request entity and the syntax of the request entity is correct but was unable to process the contained instructions.
500	Internal Server Error	IETF RFC 7231 [12]	Server is unable to process the request. Retrying the same request later might eventually succeed. In the present document, this code is also used as "catch-all" code for server errors.
503	Service Unavailable	IETF RFC 7231 [12]	Server is unable to process the request due to internal overload.

In general, error response codes used for application errors should be mapped to the most similar HTTP error status code. If no such code is applicable, one of the codes 400 (Bad Request, for client errors) or 500 (Internal Server Error, for server errors) should be used. Implementations may use additional error response codes on top of the ones listed in table 4.3.5.5-1, as long as they are valid HTTP response codes; and should include a ProblemDetails structure in the payload body as defined in clause 4.3.5.2. A list of all valid HTTP response codes and their specification documents can be obtained from the HTTP status code registry [i.4].

4.4   Common data types

4.4.1   Structured data types

4.4.1.1   Introduction

This clause defines data structures that are referenced from data structures in multiple interfaces.

4.4.1.2   Type: Object

An object contains structured data, and shall comply with the provisions of clause 4 of IETF RFC 7159 [11].

4.4.1.3   Type: Link

This type represents a link to a resource. It shall comply with the provisions defined in table 4.4.1.3-1.

Table 4.4.1.3-1: Definition of the Link data type

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI of the referenced resource

4.4.1.4   Type: KeyValuePairs

This type represents a list of key-value pairs. The order of the pairs in the list is not significant. In JSON, a set of key-value pairs is represented as an object. It shall comply with the provisions defined in clause 4 of IETF RFC 7159 [11]. In the following example, a list of key-value pairs with four keys ("aString", "aNumber", "anArray" and "anObject") is provided to illustrate that the values associated with different keys can be of different type.

EXAMPLE:

```
{
    "aString" : "ETSI NFV SOL",
    "aNumber" : 0.03,
    "anArray" : [1,2,3],
    "anObject" : {"organization" : "ETSI", "isg" : "NFV", workingGroup" : "SOL"}
```

}

4.4.1.5 Type: VnfInstanceSubscriptionFilter

This type represents subscription filter criteria to match VNF instances. It shall comply with the provisions defined in table 4.4.1.5-1.

Table 4.4.1.5-1: Definition of the VnfInstanceSubscriptionFilter data type

Attribute name	Data type	Cardinality	Description
vnfdIds	Identifier	0..N	If present, match VNF instances that were created based on a VNFD identified by one of the vnfdId values listed in this attribute. See note 1.
vnfProductsFromProviders	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products from certain providers. See note 1.
>vnfProvider	String	1	Name of the VNF provider to match.
>vnfProducts	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products with certain product names, from one particular provider.
>>vnfProductName	String	1	Name of the VNF product to match.
>>versions	Structure (inlined)	0..N	If present, match VNF instances that belong to VNF products with certain versions and a certain product name, from one particular provider.
>>>vnfSoftwareVersion	Version	1	Software version to match.
>>>vnfdVersions	Version	0..N	If present, match VNF instances that belong to VNF products with certain VNFD versions, a certain software version and a certain product name, from one particular provider.
vnfInstanceIds	Identifier	0..N	If present, match VNF instances with an instance identifier listed in this attribute. See note 2.
vnfInstanceNames	String	0..N	If present, match VNF instances with a VNF Instance Name listed in this attribute. See note 2.
NOTE 1: The attributes "vnfdIds" and "vnfProductsFromProviders" are alternatives to reference to VNF instances that are based on certain VNFDs in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			
NOTE 2: The attributes "vnfInstanceIds" and "vnfInstanceNames" are alternatives to reference to particular VNF instances in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

4.4.2 Simple data types and enumerations

4.4.2.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in multiple interfaces.

4.4.2.2 Simple data types

Table 4.4.2.2-1: Simple data types

Type name	Description
Identifier	An identifier with the intention of being globally unique. Representation: string of variable length.
IdentifierInVnfd	An identifier that is unique within a VNF descriptor. Representation: string of variable length.
IdentifierInVim	An identifier maintained by the VIM or other resource provider. It is expected to be unique within the VIM instance. Representation: string of variable length.
IdentifierInVnf	An identifier that is unique for the respective type within a VNF instance, but that need not be globally unique. Representation: string of variable length.
IdentifierLocal	An identifier that is unique within a limited local scope other than above listed identifiers, such as within a complex data structure or within a request-response pair. Representation: string of variable length.
DateTime	Date-time stamp. Representation: String formatted according to IETF RFC 3339 [3].

Type name	Description
Uri	String formatted according to IETF RFC 3986 [4].
Boolean	The Boolean is a data type having two values (true and false).
MacAddress	A MAC address. Representation: string that consists of groups of two hexadecimal digits, separated by hyphens or colons.
IpAddress	An IPV4 or IPV6 address. Representation: In case of an IPV4 address, string that consists of four decimal integers separated by dots, each integer ranging from 0 to 255. In case of an IPV6 address, string that consists of groups of zero to four hexadecimal digits, separated by colons.
Version	A Version. Representation: string of variable length.

4.4.2.3   Enumerations

This clause is empty in the present document.

4.5   Authorization of API requests and notifications

4.5.1   Introduction

The ETSI NFV MANO APIs are only allowed to be accessed by authorized consumers. Handling of authorization differs between making an API call, and sending a notification. In the former case, OAuth 2.0 is used. In the latter case, OAuth 2.0 or HTTP Basic authentication is used, and the flows differ from those used in the former case. Alternatively, a solution based on public/private key pair as authentication alternative to client identifier/password is also allowed.

The following terms (set in italics below) are used as defined by IETF RFC 6749 [9]: *client, resource server, authorization server, token endpoint, access token*. The description below is based on the "client credentials" grant type as defined by IETF RFC 6749 [9].

For API calls, the producer functional block of an API in NFV terms corresponds to the "*resource server*", and the consumer functional block of an API corresponds to the "*client*" as defined by IETF RFC 6749 [9]. For sending a notification, these roles are reversed: The producer (notification sender) corresponds to the "*client*", and the consumer (notification receiver) corresponds to the "*resource server*".

Before invoking an HTTP method on a REST resource provided by a *resource server*, a functional block (referred to as "*client*" from now on) first obtains authorization from another functional block fulfilling the role of the "*authorization server*". The present document makes no assumption about which functional block in the architecture plays the role of the *authorization server*. It is however assumed that the address of the *token endpoint* exposed by the *authorization server* and further specified in the clauses below is provisioned to the *client* together with additional authorization-related configuration parameters, such as valid client credentials. The *client* requests an *access token* from the *token endpoint*. As part of the request, it authenticates towards the *authorization server* by presenting its client credentials, consisting of client identifier and client password. The *authorization server* responds with an *access token* which the *client* will present to the *resource server* with every HTTP method invocation. An *access token* represents a particular access right (defining the particular set of protected resources to access in a particular manner) with a defined duration. The token is opaque to the *client*, and can typically be used by the *authorization server* and the *resource server* as an identifier to retrieve authorization information, such as information that identifies the client, its role and access rights. An *access token* expires after a certain time, or can be revoked. If that happens, the *client* can try to obtain a new *access token* from the *authorization server*.

In order to ensure that no third party can eavesdrop on sensitive information such as client credentials or access tokens, HTTP over TLS is used to protect the transport. HTTP over TLS enables TLS certificate authentication as an alternative to token-based approach. Moreover, if mutual authentication using TLS protocol is used, then the producer/server is authenticated to the consumer/client, but also the consumer/client is authenticated by the producer/server at the same time. To facilitate this mutual authentication, the server shall request a client certificate. This can be done as described in IETF RFC 5246 [6], including the optional CertificateRequest from server to client.

4.5.2   Flows (informative)

4.5.2.0   General

This clause outlines several alternative methods for authentication and authorization. Clause 4.5.2.1 presents an approach for authorizing API requests using OAuth 2.0 access tokens. Clause 4.5.2.1a describes an alternative method for authorization of API requests using TLS certificates. Clauses 4.5.2.2 and 4.5.2.3 outline a method to authorize notifications using basic authentication and OAuth2.0based approaches respectively. Finally, authorization of notifications using TLS certificates is presented in clause 4.5.2.4.

4.5.2.1   Authorization of API requests using OAuth 2.0 access tokens

The flow below illustrates the authorization of API requests that the API consumer sends to the API producer.

>   NOTE: Typical choices for the implementation of the authorization server include the authorization server as a component of the API producer, or as an external component.

Preconditions:

- Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.1-1.

- Authorization server is configured with the authorization policy and access rights against the client credentials.
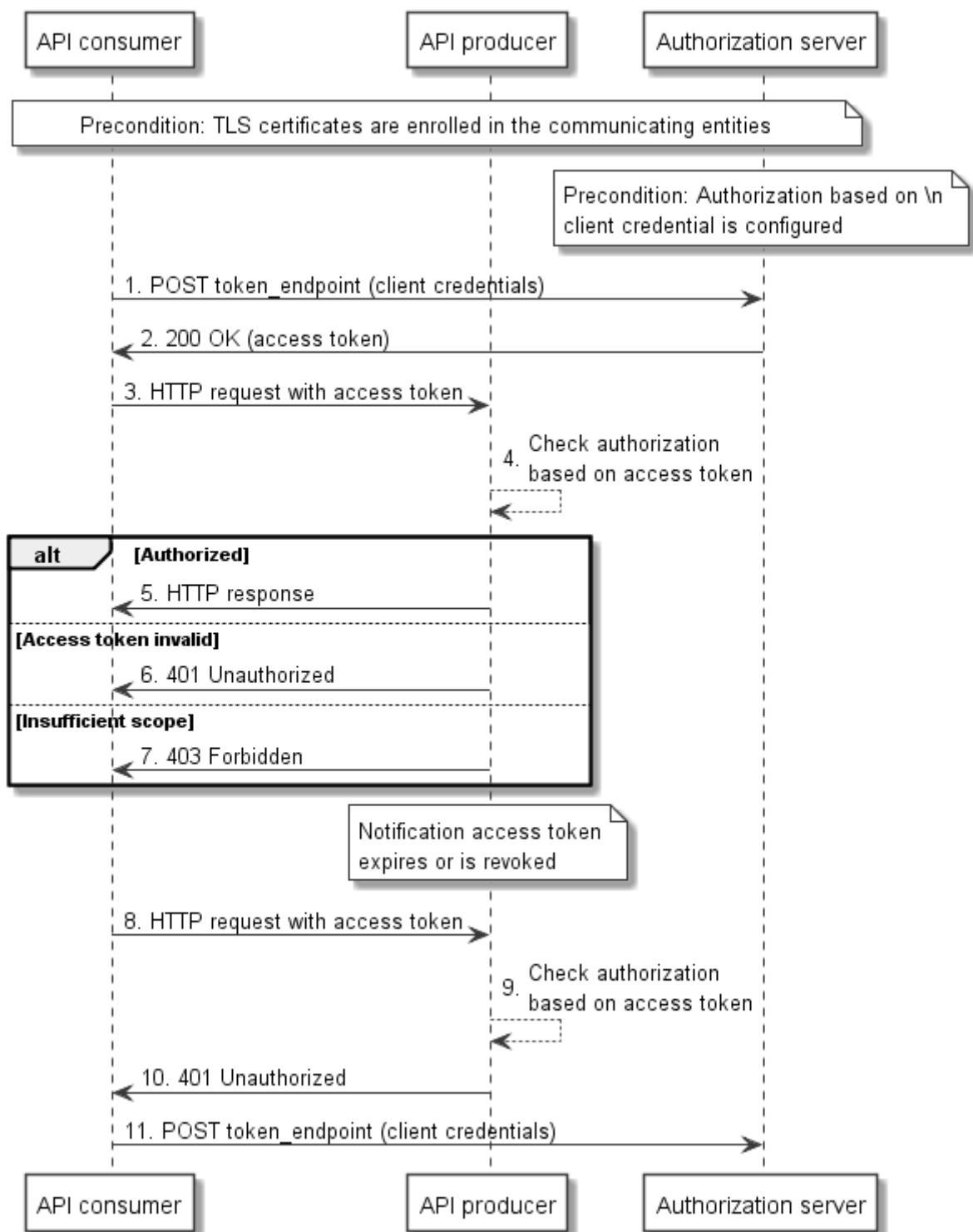
Figure 4.5.2.1-1: Authorization of API requests using OAuth 2.0 access tokens

The flow consists of the following steps:

1. To obtain an access token, the API consumer sends a POST request to the token endpoint of the authorization server and includes its client credentials.

2. The authorization server responds to the API consumer with an access token, and possibly additional information such as expiry time.

3. The API consumer sends an HTTP request to a resource provided by the API producer and includes the received access token.

4.   The API producer checks the token for validity. This assumes that it has received information about the valid access tokens, and additional related information (e.g. time of validity, client identity, client access rights) from the authorization server. Such exchange is outside the scope of the present document, and assumed to be trivial if deployments choose to include the authorization server as a component into the API producer.

5.   In case the token is valid and refers to access rights that allow accessing the actual resource with the actual request and its parameters, the API producer returns the HTTP response.

6.   In case the token is invalid or expired, the API producer returns a "401 Unauthorized" response.

7.   In case the access rights are insufficient to access the resource or to use the parameters, the API producer returns a "403 Forbidden" response.

8.   The API consumer sends an HTTP request to the API producer and includes in the request the access token.

9.   The API producer checks the token for validity, and establishes that it has expired, or has been revoked by the authorization server using means outside the scope of the present document.

10.   The API producer responds with a "401 Unauthorized" response, indicating that the access token is invalid.

11.   The API consumer attempts to obtain a new access token, as defined in step 3. This may eventually succeed or fail, depending on whether access is allowed for that API consumer any longer.

NOTE: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.1a   Authorization of API requests using TLS certificates

As an alternative to the authorization using OAuth 2.0 access tokens, authentication and authorization is defined herein based on TLS certificates, applying the IETF RFC 5246 [6]. To facilitate mutual authentication during TLS tunnel setup process, the server requests a client certificate as described in section 7.4.4 in the IETF RFC 5246 [6].

Preconditions:

•   Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.1a-1.

•   Authorization server is configured with the authorization policy and access rights against the certificates.
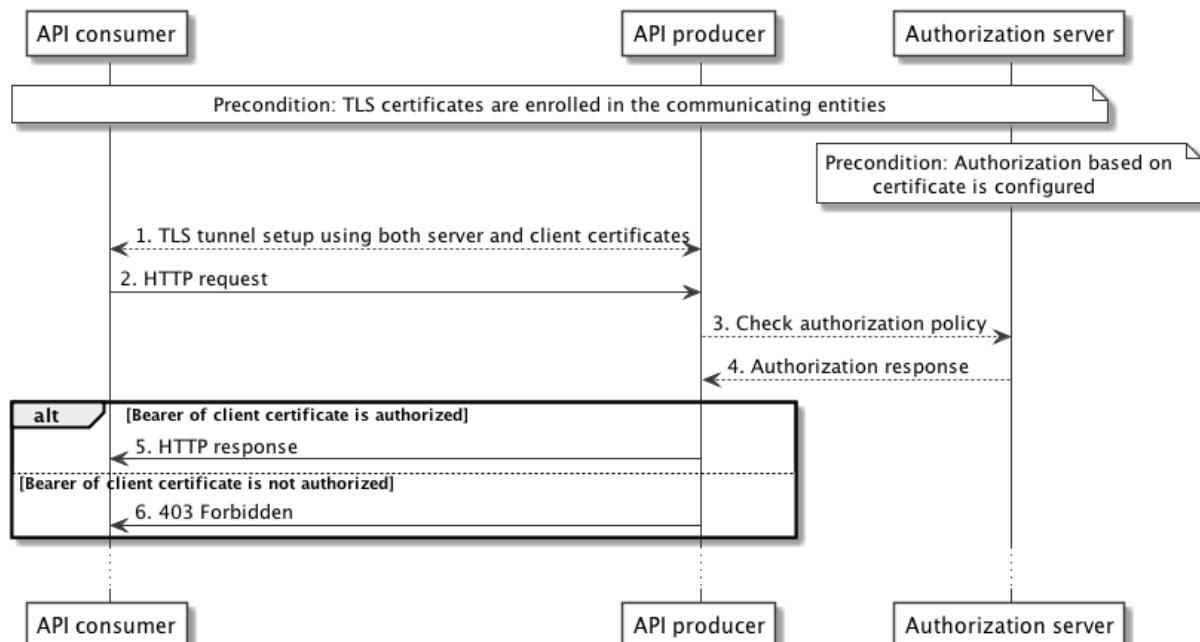
Figure 4.5.2.1a-1: Authorization of API requests using TLS certificates

The flow consists of the following steps:

1) The API consumer initiates the TLS tunnel setup process with the API producer. During the tunnel setup process the API producer sends its certificate to API consumer and obtains the certificate from the API consumer by including the CertificateRequest message specified in IETF RFC 5246 [6]. This ensures the mutual authentication between the consumer and the producer.

2) API consumer further sends the HTTP request for a resource over the TLS tunnel.

3) API producer now checks for the authorization information from the authorization server based on the API consumer client certificate.

4) Authorization server checks its policy and sends the response to the API producer.

5) If the API consumer is authorized, then the API producer sends the response related to the requested resource.

6) If the API consumer is unauthorized, then the API producer sends "403 Forbidden" response to the API consumer.

NOTE 1: Steps 3 and 4 are outside the scope of the present document. However, typical implementations can use the certificates in such a way that the API producer verifies the certificate of the API consumer and extracts the subject name from the certificate. This information will be sent to the authorization server in order to check the authorization. In a response, the authorization server will send the associated client profile that contains the access rights.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.2   Authorization of notifications using the HTTP Basic authentication scheme

The flow below illustrates the authorization of notifications that the API producer sends to the API consumer based on the HTTP Basic authentication scheme (see IETF RFC 7617 [16]). In this flow, no authorization server is needed.
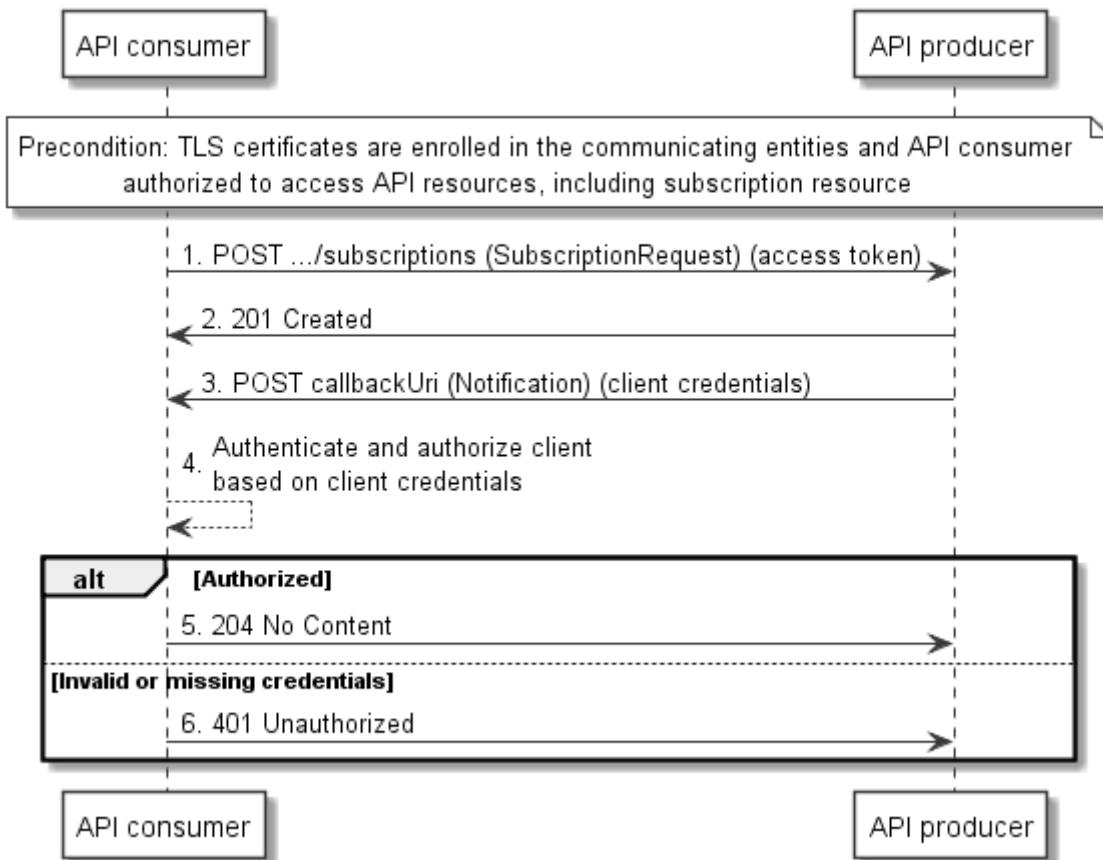
Figure 4.5.2.2-1: Authorization of notifications using the HTTP Basic authentication scheme

It is a precondition for this flow that the API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1. Additionally, to ensure secure communication, it is a precondition that the TLS certificates are enrolled in the communicating entities.

The flow consists of the following steps:

1. The API consumer sends a request to create a new subscription resource to the API producer and includes in the request a valid access token to prove that it is authorized to access the API. Also, it includes in the subscription client credentials that the API producer can use to authenticate towards the API consumer when subsequently sending notifications. Note that these credentials are typically different from the client credentials used in the flow in clause 4.5.2.1.

2. The API producer creates the subscription resource and responds with "201 Created".

3. The API producer sends an HTTP POST request with a notification to the callback URI registered by the API consumer during subscription, and includes the client credential in the request to authenticate.

4. The API consumer checks the credentials against the information it has sent in step 1.

5. In case the credentials are valid, the API consumer returns a "204 No Content" HTTP response to indicate successful delivery of the notification.

6. In case the credentials are invalid, the API consumer returns a "401 Unauthorized" response.

NOTE: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.3   Authorization of notifications using OAuth 2.0 access tokens

The flow below illustrates the authorization of notifications that the API producer sends to the API consumer using OAuth 2.0. In this flow, the authorization server can be a different entity than the authorization server in clause 4.5.2.1.

NOTE: Typical choices for the implementation of the authorization server include the authorization server as a component of the API consumer, or as an external component.
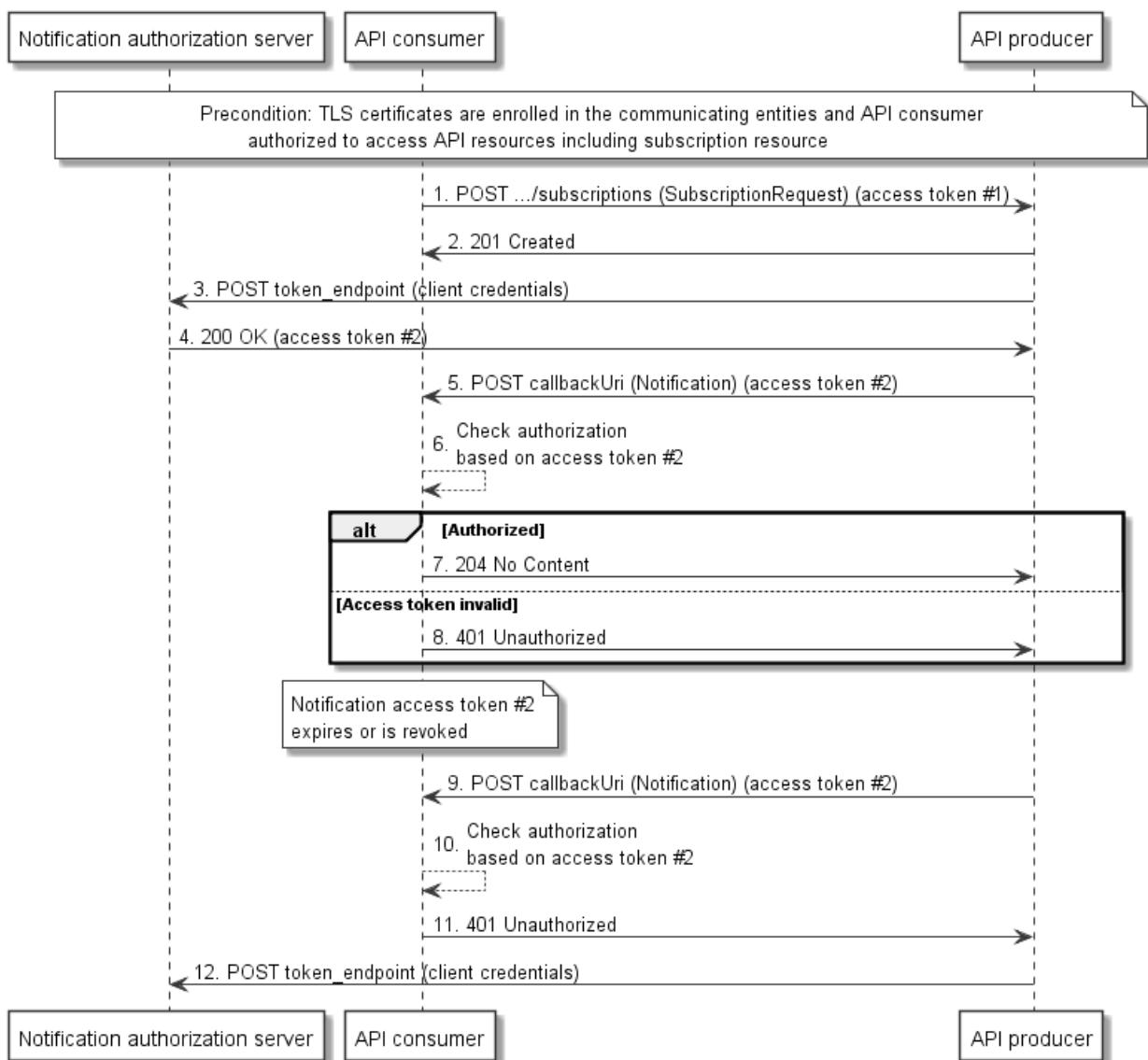
Figure 4.5.2.3-1: Authorization of notifications using OAuth 2.0 access tokens

It is a precondition for this flow that the API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1. Additionally, to ensure secure communication, it is a precondition that the TLS certificates are enrolled in the communicating entities.

The flow consists of the following steps:

1. The API consumer sends a request to create a new subscription resource to the API producer and includes in the request a valid access token #1 to prove that it is authorized to access the API. Also, it includes in the subscription request parameters that the API producer can use to obtain authorization to send notifications to the API consumer, such as client credentials and a token endpoint. Note that these are typically different from the credentials and token endpoint used in the flow in clause 4.5.2.1.

2. The API producer creates the subscription resource and responds with "201 Created".

3. Subsequently, and prior to sending any notification to the API consumer, the API producer obtains authorization to do so by requesting an access token from the authorization server, using the end point and notification client credentials that were sent in the subscription request, or provisioned otherwise.

4. The authorization server responds to the API producer with an access token, hereafter called access token #2 and possibly additional information such as expiry time.

5. The API producer sends an HTTP POST request with a notification to the callback URI registered by the API consumer during subscription, and includes the received access token #2.

6. The API consumer checks the token for validity. This assumes that it has received information about the valid access tokens, and additional related information (e.g. time of validity, client identity, client access rights) from the authorization server. Such exchange is outside the scope of the present document, and assumed to be trivial if deployments choose to include the authorization server as a component into the API consumer.

7. In case the token #2 is valid, the API consumer returns a "204 No Content" HTTP response to indicate successful delivery of the notification.

8. In case the token #2 is invalid or expired, the API consumer returns a "401 Unauthorized" response.

9. The API producer sends another notification in an HTTP POST request to the API consumer and includes in the request the access token #2.

10. The API consumer checks the token #2 for validity, and establishes that it has expired, or has been revoked by the authorization server using means outside the scope of the present document.

11. The API consumer responds with a "401 Unauthorized" response, indicating that the access token #2 is invalid.

12. The API producer attempts to obtain a new access token. This may eventually succeed or fail, depending on whether access is allowed for that API producer any longer.

NOTE: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

4.5.2.4   Authorization of notifications using TLS certificates

The flow in Figure 4.5.2.4-1 illustrates the authorization of notifications that the API producer sends to the API consumer using TLS certificates.

Preconditions:

- Certificates are enrolled in the communicating entities as shown in the figure 4.5.2.4-1.

- The API consumer is authorized to access the "subscriptions" resource provided by the API producer, using the procedure illustrated in clause 4.5.2.1 or 4.5.2.1a.
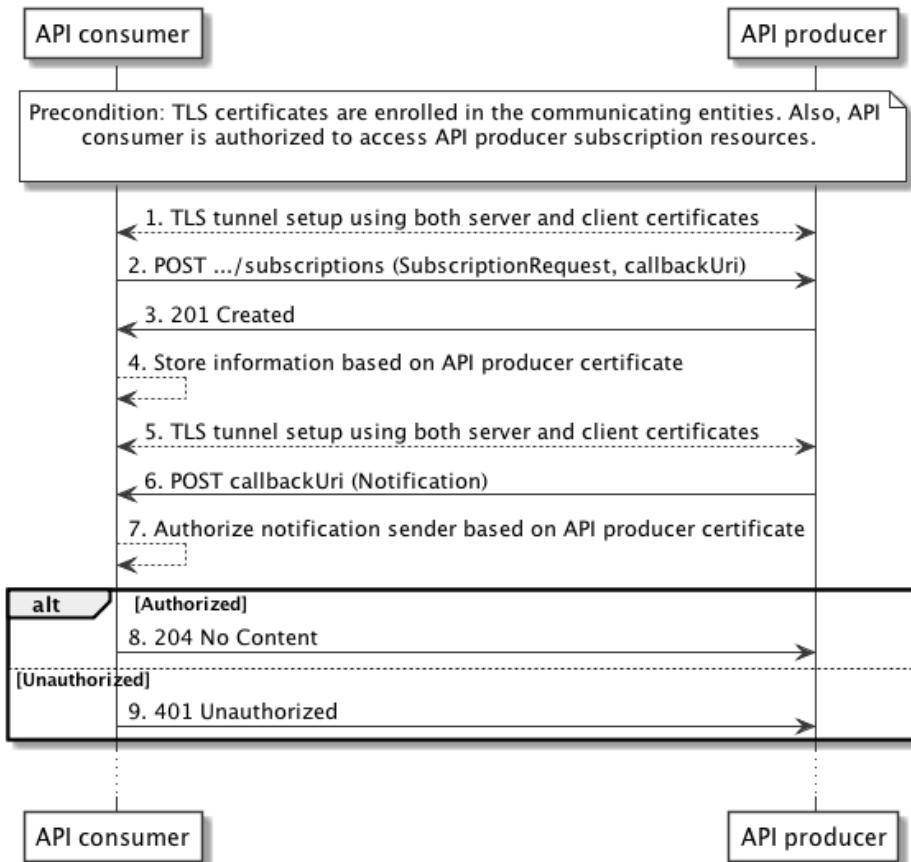
Figure 4.5.2.4-1: Authorization of notifications using TLS certificates

The flow consists of the following steps:

1. The API consumer initiates the TLS tunnel setup process with the API producer. During the tunnel setup process the API producer obtains the certificate from the API consumer. This ensures the mutual authentication between the consumer and the producer.

2. The API consumer sends a request to create a new subscription resource to the API producer. The API producer can authenticate and authorize this request based on the API consumer certificate as illustrated in clause 4.5.2.1a. The request also includes the callbackURI where the notification will be sent in future.

3. The API producer creates the subscription resource and responds with "201 Created".

4. The API consumer now stores the relevant information of the API producer's certificate in association with the requested notification subscription.

5. The API producer initiates the TLS tunnel with the API consumer whenever there is a notification to send. During the tunnel setup process the API consumer sends its certificate to API producer and obtains the client certificate from the API producer. This ensures the mutual authentication between the consumer and the producer.

6. The API producer sends the notification over the established TLS tunnel.

7. API consumer can now verify whether this sender is allowed to send this notification by matching the sender's certificate information with the previously stored information at step 4.

8. In case is the API producer is authorized to send a notification, then the API consumer sends a "204 No Content" response to indicate successful delivery of the notification.

9. In case if the API producer is not authorized to send a notification, the API consumer returns a "403 Forbidden" response.

NOTE 1: Steps 4 and 7 are outside the scope of the present document. However, typical implementation can use the certificates in such a way that the API consumer verifies the certificate of the API producer and extract subject name from the certificate. This information is used in order to check the authorization at the API consumer.

NOTE 2: All the communication presented in this flow diagram is done over encrypted tunnel using TLS as described in clause 4.2.

NOTE 3: It is assumed that the API producer uses the same certificate for both the client and server role.

4.5.3   Specification

4.5.3.1   Introduction

OAuth 2.0 provides a framework for authorization of web applications that has multiple modes and options. This clause profiles the framework for use in the context of the Ve-Vnfm reference point. Clause 4.5.3.2 specifies the general mechanism. Two different uses of the general mechanism, actually for API requests and for sending notifications, are defined in clauses 4.5.3.3 and 4.5.3.4.

4.5.3.2   General mechanism

For all requests to an API defined in the present document, and for all notifications sent via such an API, authorization as defined below shall be used. Requests and notifications without authorization credentials shall be rejected.

To allow the *client* to obtain an access token, the *authorization server* shall expose a *token endpoint* that shall comply with the provisions defined by the OAuth 2.0 specification for the *client credentials* grant type (see IETF RFC 6749 [9]). A *client* shall use the access token request and response according to this grant type, as defined by IETF RFC 6749 [9], to obtain an *access token* for access to the REST resources defined by the present specification. The content of the *access token* is out of the scope of the present document; however, it shall not be possible for an attacker to easily guess it. The *access token* shall be a string. The set of allowed characters is defined in IETF RFC 6749 [9].

A *client* that invokes an HTTP request towards a resource defined by one of the APIs of the present document shall include the *access token* as a bearer token in every HTTP method in the "Authorization" HTTP header, as defined by IETF RFC 6750 [10]. A *resource server* that receives an HTTP request with an invalid *access token*, or without an *access token,* shall reject the request, and shall signal the error in the HTTP response according to the provisions for the error codes and the "WWW-Authenticate" response HTTP header as defined by IETF RFC 6750 [10].

A *client* that receives a rejection of an *access token* may obtain a new *access token* from the *token endpoint* of the *authorization server*, and retry the request.

As an alternative to OAuth 2.0 access tokens, certificates, as defined by TLS 1.2 in IETF RFC 5246 [6], can be used to facilitate the authentication and authorization between client and the server.

4.5.3.3   Authorizing API requests

A consumer of an API that wishes to issue HTTP requests towards resources provided by that API shall act as a *client* according to clause 4.5.3.2 to obtain an access token, and shall include this access token in every HTTP request, as defined in clause 4.5.3.2. The respective API producer shall act as a *resource server* as defined in clause 4.5.3.2.

Alternatively, API requests can be authorized based on TLS certificates.

These two different alternatives are listed in the following:

1)   API consumer passes access token when accessing a resource provided by API producer. API producer checks authorization based on access token. Access token can be obtained from the authorization server based on client ID and password.

2)   API consumer accesses a resource provided by API producer using TLS tunnel where both server and client certificates are used to establish the secure tunnel. API producer checks authorization based on client's TLS certificate. The client's TLS certificate is obtained during the TLS handshake.

4.5.3.4   Authorizing the sending of notifications

The procedure defined in clause 4.5.2 allows an API consumer to obtain authorization to perform API requests towards the API producer, including subscription requests. For sending the actual notifications matching a subscription, the API producer needs to obtain separate authorization to actually *send* the notification to the API consumer.

If an API consumer requires the API producer to authorize for sending notifications to that API consumer, it shall include in the subscription request a data structure that defines the authorization requirements, as defined in table 4.5.3.4-1.

Table 4.5.3.4-1: Definition of the SubscriptionAuthentication data type

Attribute name	Data type	Cardinality	Description
authType	Enum (inlined)	1..N	Defines the types of Authentication/Authorization. The API consumer is willing to accept when reeiving a notification. Permitted values: - BASIC: In every HTTP request to the notification endpoint, use HTTP Basic authentication with the client credentials. - OAUTH2_CLIENT_CREDENTIALS: In every HTTP request to the notification endpoint, use an OAuth 2.0 Bearer token, obtained using the client credentials grant type. - TLS_CERT: Every HTTP request to the notification endpoint is sent over a mutually authenticated TLS session. i.e. not only server is authenticated, but also the client is authenticated during the TLS tunnel setup.
paramsBasic	Structure (inlined)	0..1	Parameters for authentication/authorization using BASIC. Shall be present if authType is "BASIC" and the contained information has not been provisioned out of band. Shall be absent otherwise.
>userName	String	0..1	Username to be used in HTTP Basic authentication. Shall be present if it has not been provisioned out of band.
>password	String	0..1	Password to be used in HTTP Basic authentication. Shall be present if it has not been provisioned out of band.
paramsOauth2Client Credentials	Structure (inlined)	0..1	Parameters for authentication/authorization using OAUTH2_CLIENT_CREDENTIALS. Shall be present if authType is "OAUTH2_CLIENT_CREDENTIALS" and the contained information has not been provisioned out of band. Shall be absent otherwise.
>clientId	String	0..1	Client identifier to be used in the access token request of the OAuth 2.0 client credentials grant type. Shall be present if it has not been provisioned out of band. See note.
>clientPassword	String	0..1	Client password to be used in the access token request of the OAuth 2.0 client credentials grant type. Shall be present if it has not been provisioned out of band. See note.
>tokenEndpoint	Uri	0..1	The *token endpoint* from which the access token can be obtained. Shall be present if it has not been provisioned out of band.
NOTE: The clientId and clientPassword passed in a subscription shall not be the same as the clientId and clientPassword that are used to obtain authorization for API requests. Client credentials may differ between subscriptions. The value of clientPassword should be generated by a random process.			

If the value of "authType" is "OAUTH2_CLIENT_CREDENTIALS":

- The API producer shall, prior to sending any notification, obtain an *access token* from the *token endpoint* using the OAuth 2.0 client credentials grant type as defined in IETF RFC 6749 [9]. The API consumer should include expiry information with the token response.

- The API producer shall include that *access token* as a Bearer token in every POST request that sends a notification (according to IETF RFC 6750 [10]).

- If the *access token* is expired, the API consumer shall reject the notification. In that case, the API producer shall obtain a new *access token*, and repeat sending the notification.

- If the token expiry time is known to the API producer, it may obtain proactively a new access token.

If the value of "authType" is "BASIC":

- The API producer shall pass its client credentials in every POST request that sends a notification, as defined in IETF RFC 7617 [16].

If the value of "authType" is "TLS_CERT":

- The API producer (client) shall use its TLS certificate to create a mutually authenticated TLS session with the API consumer (server) and further the API consumer will do the authorization based on the API producer's certificate.

4.5.3.5   Client roles

An *access token* allows the API producer to identify information about the *client* that has obtained the access token, such as client identity, client role or client access rights. By having this property, *access tokens* can be used as a means to distinguish between different roles (and consequently different access rights) to the same set of resources.

The mechanism for this works as follows: By means out of scope of the present document, the role of the client identified by a particular client identifier is provisioned to the authorization server. When that client obtains an access token, it sends its client identifier and client password to the authorization server. The authorization sever can obtain the role of the client by evaluating the data that were provisioned for the client identifier, and associate that information to the access token. By means out of scope of the present document, that association is shared with the API producer. This enables the API producer to detect the role based on the access token.

In ETSI NFV, certain interfaces are exposed on multiple different reference points, i.e. the same interface is exposed to different consumer functional blocks. Depending on the consumer block that originates an HTTP request, not all resources/HTTP methods/request and parameters might be available. From the point of view of the producer functional block, this can be seen as consumers acting in different roles when accessing a particular interface, such as the VNF LCM interface.

Implementations may use the OAuth *access token* to differentiate between these cases, assuming that an *access token* can determine whether a consumer functional block acts in the role of the VNFM, the VNF or the EM. This assumes that the role of the consumer functional block is bound to its client credentials. The means of creating this binding is out of scope of the present document (e.g. a configuration step or policy).

As an alternative mechanism, the client role can be bound to its certificate. The mechanism for this works as follows: By means out of scope of the present document, the client is identified by a particular client subject name that is extracted from its certificate. This subject name is then provided to the authorization server in order to get the associated role of that particular client. By means out of scope of the present document, the authorization server is preconfigured to have this association between the client subject name and the role.

4.5.3.6   Negotiation of authorization method

4.5.3.6.1   Authorization of API requests

The following provisions apply to the support of the authorization methods defined in the present document for the authorization of API requests:

- The API producer shall support checking the authorization of API requests it receives based on an OAuth 2.0 access token as defined in clause 4.5.3.3, and should support checking the authorization of API requests it receives based on TLS certificates as defined in clause 4.5.3.3.

- The API consumer shall support the authorization of API requests it sends by including an OAuth 2.0 bearer token in the request as defined in clause 4.5.3.3, and should support the authorization of API requests it sends by providing its client certificate to the API producer during TLS tunnel setup as defined in clause 4.5.3.3.

When performing and authorizing an API request, API consumer and API producer shall use the following procedure to negotiate the authorization method to use if the API consumer supports both the authorization based on OAuth 2.0 and the authorization based on TLS certificates.
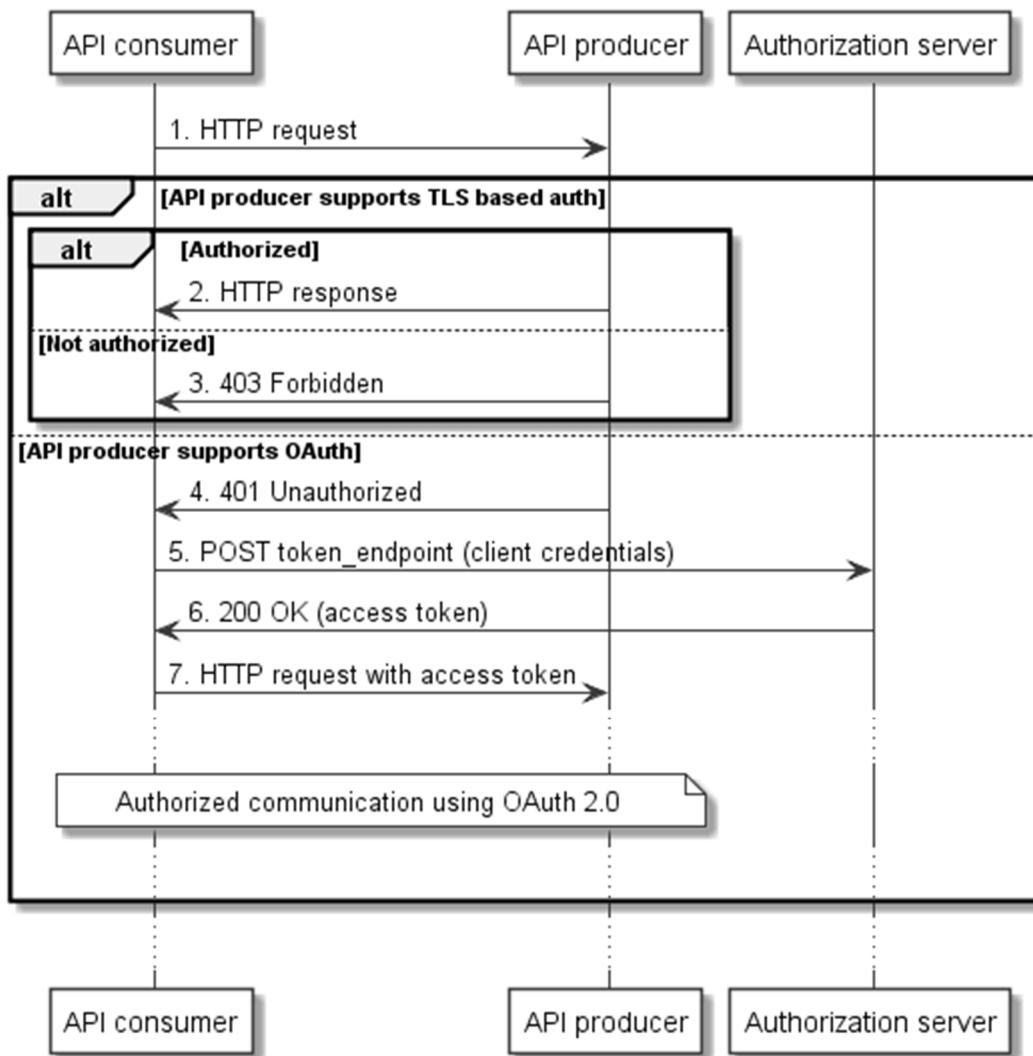
Figure 4.5.2.1-1: Negotiation of authorization method to use for API requests

1. The API consumer shall send an HTTP request to the API producer without an access token.

2. If the API producer supports the authorization based on TLS certificates and the API consumer is authorized, it shall return the HTTP response to fulfil the request. Subsequent communication between API consumer and API producer shall use the authorization based on TLS credentials.

3. If the API producer supports the authorization based on TLS certificates and the API consumer is not authorized, it shall return a 403 Forbidden response.

4. If the API producer does not support the authorization based on TLS certificates it shall return a 401 Unauthorized response.

5. Once it has received the 401 Unauthorized response, the API consumer shall subsequently request an access token from the authorization server, according to clause 4.5.3.2.

6. The authorization server shall respond with an access token according to clause 4.5.3.2.

7. The API consumer shall subsequently retry the HTTP request with the access token included as a bearer token according to clause 4.5.3.2.

Subsequent authorized communication between API consumer and API producer shall take place as defined in clause 4.5.3.2 (see also the flow in clause 4.5.2.1, starting at step 4).

4.5.3.6.2 Authorization of notification requests

The following provisions apply to the support of the authorization methods defined in the present document for the authorization of notification requests.

- The API consumer shall support checking the authorization of notification requests it receives based on an OAuth 2.0 access token as defined in clause 4.5.3.4. Further, the API producer should support checking the authorization of notification requests it receives based on HTTP Basic authentication as defined in clause 4.5.3.4, and based on TLS certificates as defined in clause 4.5.3.4.

- The API producer shall support the authorization of notification requests it sends by including an OAuth 2.0 bearer token in the request as defined in clause 4.5.3.4. Further, the API producer should support the authorization of notification requests it sends by providing credentials based on HTTP Basic authentication as defined in clause 4.5.3.4, and by providing its client certificate to the API producer during TLS tunnel setup as defined in clause 4.5.3.4.

When performing and authorizing a notification request, API consumer and API producer shall use the following procedure to negotiate the authorization method to use:

1) The API consumer shall signal in the subscription the authorization methods it accepts for notifications related to that particular subscription.

2) If only none of the methods signalled is supported by the API producer, the API producer shall reject the subscription with "422 Unprocessable Entity", and shall include in the payload body a ProblemDetails structure which shall provide the reason for the rejection in the "details" attribute.

3) Otherwise, the API producer shall select one of the authorization methods that was signalled in the subscription, and shall use that method for the authorization of notifications it sends based on that subscription.

5 VNF Lifecycle Management interface

5.1 Description

This interface allows the VNF/EM to invoke VNF lifecycle management operations of VNF instances towards the VNFM, and to subscribe to notifications regarding VNF lifecycle changes provided by the VNFM.

The operations defined for VNF through this interface are:

- Query VNF
- Scale VNF
- Scale VNF to Level
- Heal VNF

- Get Operation Status

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

The operations defined for EM through this interface are:

- Create VNF Identifier

- Query VNF

- Modify VNF Information

- Delete VNF Identifier

- Instantiate VNF

- Scale VNF

- Scale VNF to Level

- Change VNF Flavour

- Terminate VNF

- Heal VNF

- Operate VNF

- Change external VNF connectivity

- Get Operation Status

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

This interface also enables to invoke error handling procedures (Retry, Rollback, Cancel, Fail) on the actual VNF lifecycle management operation occurrences.

5.2   Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnflcm" shall be used to represent {apiName}. The {apiVersion} shall be set to "v1" for the present document. All resource URIs in the clauses below are defined relative to the above base URI.

Figure 5.2-1 shows the overall resource URI structure defined for the VNF lifecycle management interface.
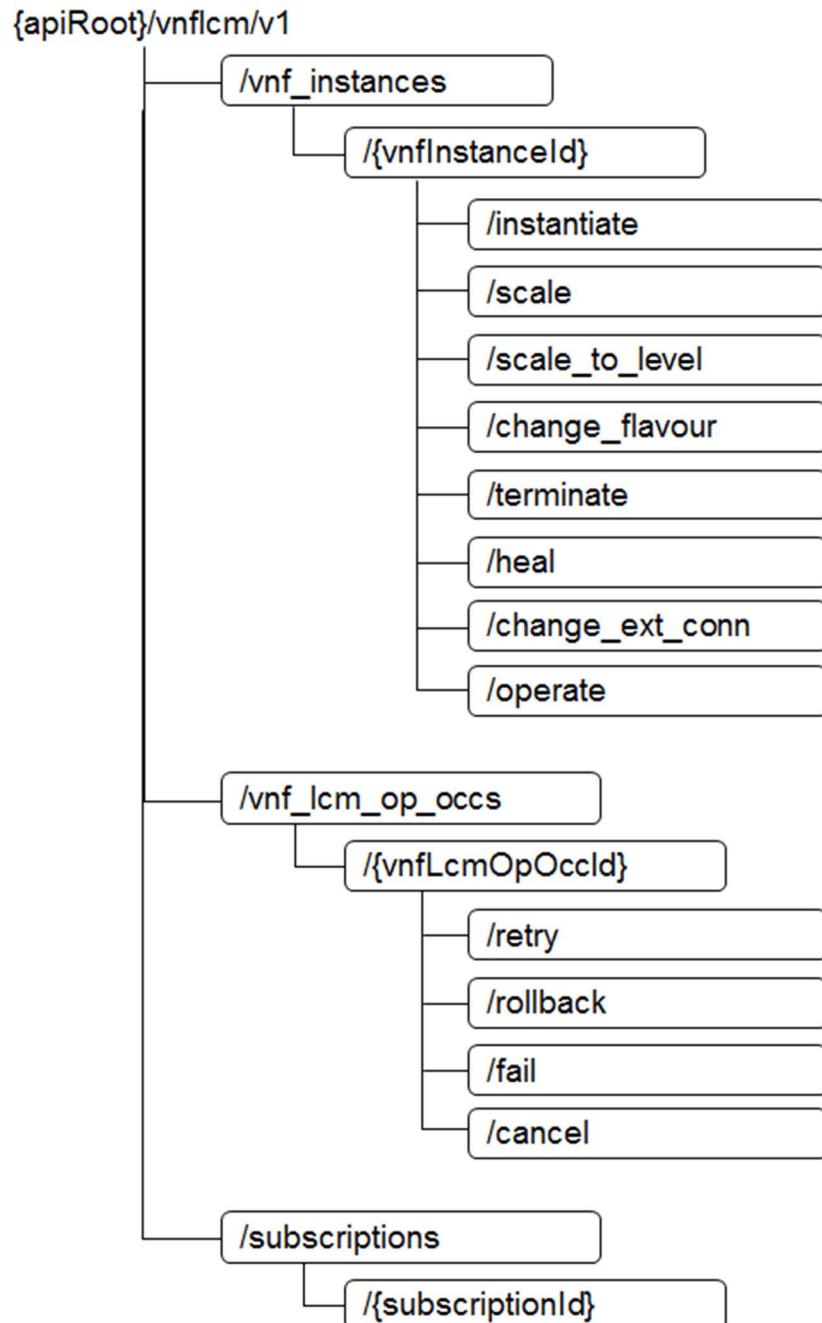
Figure 5.2-1: Resource URI structure of the VNF Lifecycle Management Interface

Table 5.2-1 lists the individual resources defined, and the applicable HTTP methods.

The VNFM shall support responding to requests for all HTTP methods on the resources in Table 5.2-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 5.2-1: Resources and methods overview of the VNF Lifecycle Management interface

Resource name	Resource URI	HTTP Method	CAT	Meaning
VNF instances	/vnf_instances	GET	M	Query multiple VNF instances
		POST	M	Create a VNF instance resource
Individual VNF instance	/vnf_instances/{vnfInstanceId}	GET	M	Read an individual VNF instance resource
		PATCH	M	Modify VNF instance information
		DELETE	M	Delete VNF instance resource
Instantiate VNF task	/vnf_instances/{vnfInstanceId}/instantiate	POST	M	Instantiate a VNF
Scale VNF task	/vnf_instances/{vnfInstanceId}/scale	POST	M	Scale a VNF instance incrementally
Scale VNF to Level task	/vnf_instances/{vnfInstanceId}/scale_to_level	POST	M	Scale a VNF instance to a target level
Change VNF flavour task	/vnf_instances/{vnfInstanceId}/change_flavour	POST	M	Change the deployment flavour of a VNF instance
Terminate VNF task	/vnf_instances/{vnfInstanceId}/terminate	POST	M	Terminate a VNF instance
Heal VNF task	/vnf_instances/{vnfInstanceId}/heal	POST	M	Heal a VNF instance
Operate VNF task	/vnf_instances/{vnfInstanceId}/operate	POST	M	Operate a VNF instance
Change external VNF connectivity task	/vnf_instances/{vnfInstanceId}/change_ext_conn	POST	M	Change the external connectivity of a VNF instance
VNF LCM operation occurrences	/vnf_lcm_op_occs	GET	M	Query multiple VNF lifecycle management operation occurrences
Individual VNF LCM operation occurrence	/vnf_lcm_op_occs/{vnfLcmOpOccId}	GET	M	Read an individual VNF lifecycle management operation occurrence resource
Retry operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/retry	POST	M	Retry a VNF lifecycle management operation occurrence
Rollback operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback	POST	M	Rollback a VNF lifecycle management operation occurrence
Fail operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail	POST	M	Mark a VNF lifecycle management operation occurrence as failed
Cancel operation task	/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel	POST	M	Cancel a VNF lifecycle management operation occurrence
Subscriptions	/subscriptions	POST	M	Subscribe to VNF lifecycle change notifications
		GET	M	Query multiple subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription resource
		DELETE	M	Terminate a subscription
Notification endpoint	(client-provided)	POST	(*)	Notify about VNF lifecycle change. (*) See note.
		GET	(*)	Test the notification endpoint. (*) see note.
NOTE:	The VNFM shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the . If the EM or VNF supports invoking the POST method on the "Subscriptions" resource towards the VNFM, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.			

5.3   Sequence diagrams (informative)

5.3.1   Flow of the creation of a VNF instance resource

This clause describes the procedure for the creation of a VNF instance resource.
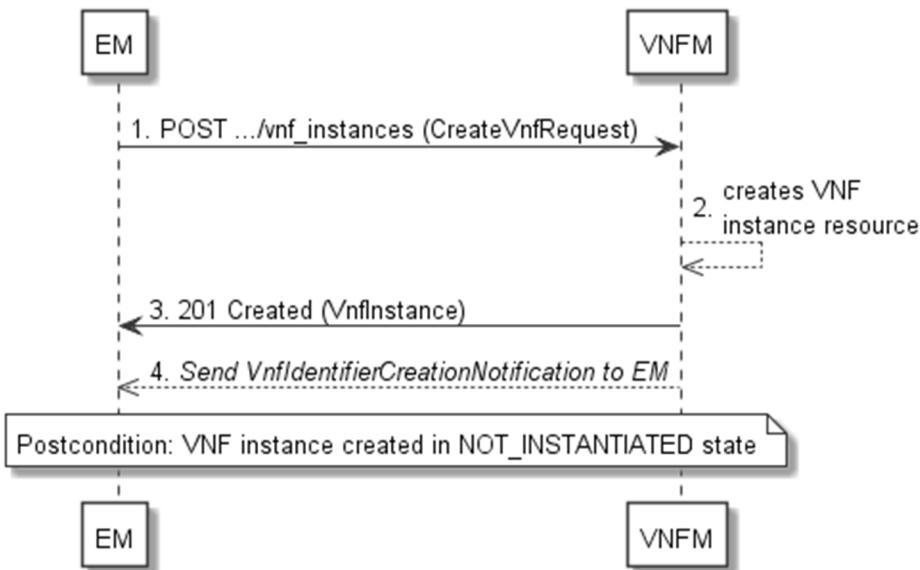
Figure 5.3.1-1: Flow of the creation of a VNF instance resource

NOTE: Due to possible race conditions, the 201 response and the VnfIdentifierCreationNotification can arrive in any order at the EM.

The procedure consists of the following steps as illustrated in figure 5.3.1-1:

1.   The EM sends a POST request to the "VNF Instances" resource including in the payload body a data structure of type "CreateVnfRequest".

2.   The VNFM creates a new VNF instance resource in NOT_INSTANTIATED state, and the associated VNF instance identifier.

3.   The VNFM returns a 201 Created response containing a representation of the VNF instance resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

4.   The VNFM sends a VNF Identifier Creation Notification (see clause 5.3.9) to the EM to indicate the creation of the VNF instance resource and the associated VNF instance identifier.

Postcondition: Upon successful completion, the VNF instance resource has been created in "NOT_INSTANTIATED" state.

5.3.2   Flow of the deletion of a VNF instance resource

This clause describes the procedure for the deletion of a VNF instance resource.
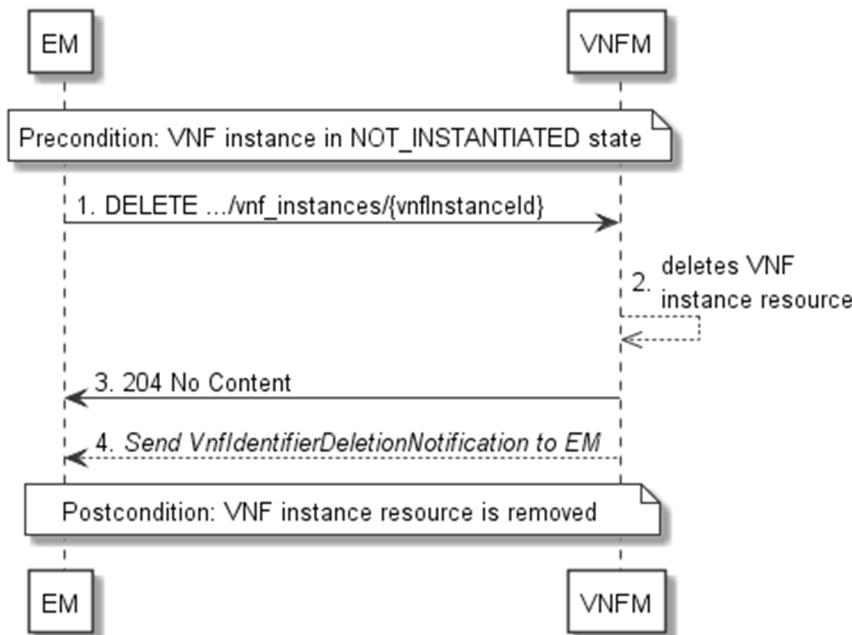
Figure 5.3.2-1: Flow of the deletion of a VNF instance resource

NOTE: Due to possible race conditions, the 204 response and the VnfIdentifierDeletionNotification can arrive in any order at the EM.

Precondition: The resource representing the VNF instance to be deleted needs to be in NOT_INSTANTIATED state.

The procedure consists of the following steps as illustrated in figure 5.3.2-1:

1.   EM sends a DELETE request to the "Individual VNF Instance" resource.

2.   The VNFM deletes the VNF instance resource and the associated VNF instance identifier.

3.   The VNFM returns a "204 No Content" response with an empty payload body.

4.   The VNFM sends to the EM a VnfIdentifierDeletionNotification to indicate the deletion of the VNF instance resource and the associated VNF instance identifier.

Postcondition: The resource representing the VNF instance has been removed from the list of VNF instance resources.

Error handling: If the VNF instance is not in NOT_INSTANTIATED state, the VNFM rejects the deletion request.

5.3.3   Flow of VNF lifecycle management operations triggered by task resources

This clause describes the general sequence for VNF Lifecycle Management operations that operate on VNF instance resource and are triggered by task resources. The flows for these operations are very similar. The differences between the individual operations are covered in the table at the end of this clause.

This flow is applicable to the following operations:

•   Instantiate VNF

•   Scale VNF

•   Scale VNF to Level

•   Change VNF flavour

•   Operate VNF

•   Heal VNF

- Change external VNF connectivity

- Terminate VNF

Figure 5.3.3-1 illustrates the general lifecycle management flow. Placeholders in this flow allow for differentiating between the operations and are marked with double angular brackets "<<…>>".

NOTE 1: The consumer can be either EM or VNF depending on the operations.
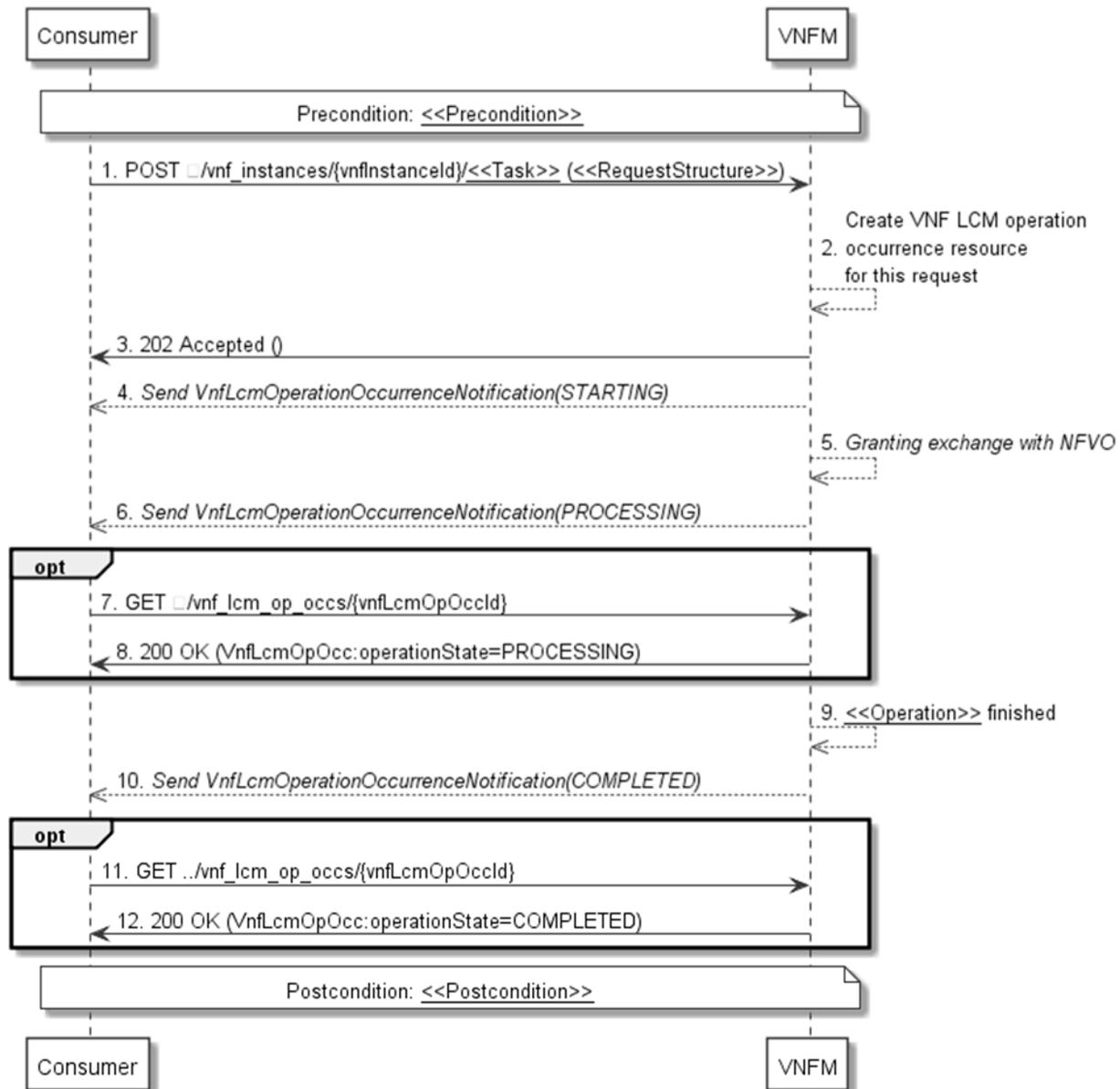
Figure 5.3.3-1: General flow of VNF lifecycle management operations triggered by task resources

NOTE 2: Due to possible race conditions, the 202 response and the "STARTING" VnfLcmOperationOccurrenceNotification can arrive in any order at the Consumer (EM/VNF).

Precondition: The precondition depends on the actual operation and is described by the template parameter <<Precondition>> in the tables below.

A VNF lifecycle operation, as illustrated in figure 5.3.3-1, consists of the following steps:

1.   The Consumer sends a POST request to the <<Task>> resource that represents the lifecycle operation to be executed on the VNF instance, and includes in the payload body a data structure of type <<RequestStructure>>. The name <<Task>> of the task resource and the <<RequestStructure>> depend on the operation and are described in the tables below.

2.   The VNFM creates a "VNF LCM operation occurrence" resource for the request.

3.   The VNFM returns a "202 Accepted" response with an empty payload body and a "Location:" HTTP header
      that points to the new "VNF LCM operation occurrence" resource, i.e. it includes the URI of that resource
      which is "…/vnf_lcm_op_occs/{vnfLcmOpOccId}". See note 2.

4.   The VNFM sends to the Consumer a lifecycle management operation occurrence notification (see clause 5.3.9)
      to indicate the start of the lifecycle management operation occurrence with the "STARTING" state. See note 2.

5.   VNFM and NFVO exchange granting information.

6.   The VNFM sends to the Consumer a VNF lifecycle management operation occurrence notification (see
      clause 5.3.9) to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state.

7.   If desired, the Consumer can poll the "VNF LCM operation occurrence" resource to obtain information about
      the ongoing operation by sending a GET request to the resource that represents the VNF LCM operation
      occurrence.

8.   In the response to that request, the VNFM returns to the Consumer information of the operation, such as the
      operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".

9.   The VNFM has finished the operation <<Operation>>.

10.   The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate
      the completion of the lifecycle management operation occurrence with the success state "COMPLETED".

11.   If desired, the Consumer can send a new GET request to the "VNF LCM operation occurrence" resource.

12.   In the response to that request, the VNFM returns to the Consumer information about the result of the
      operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: The postcondition depends on the actual operation and is described by the template parameter
<<Postcondition>> in table 5.3.3-1.

Error handling: If the VNF lifecycle management operation fails, error information is provided in the notification
message that reports the erroneous completion of the procedure, and is also available in the resource that represents the
actual VNF lifecycle management operation occurrence related to this VNF lifecycle management operation.
Table 5.3.3-1 defines how the flow described above is parameterized for the different VNF lifecycle management
operations.

Table 5.3.3-1: Parameterization of the flow for different VNF lifecycle management operations

Operation	Precondition	Task	RequestStructure	Postcondition
Instantiate VNF	VNF instance created and in NOT_INSTANTIATED state	instantiate	InstantiateVnfRequest	VNF instance in INSTANTIATED state
Scale VNF	VNF instance in INSTANTIATED state	scale	ScaleVnfRequest	VNF instance still in INSTANTIATED state and VNF was scaled
Scale VNF to Level	VNF instance in INSTANTIATED state	scale_to_level	ScaleVnfToLevel Request	VNF instance still in INSTANTIATED state and VNF was scaled
Change VNF flavor	VNF instance in INSTANTIATED state	change_flavour	ChangeVnfFlavour Request	VNF instance still in INSTANTIATED state and VNF deployment flavour changed
Operate VNF	VNF instance in INSTANTIATED state	operate	OperateVnfRequest	VNF instance still in INSTANTIATED state and VNF operational state changed
Heal VNF	VNF instance in INSTANTIATED state	heal	HealVnfRequest	VNF instance still in INSTANTIATED state
Change external VNF connectivity	VNF instance in INSTANTIATED state	change_ext_con n	ChangeExtVnfConnectiv ityRequest	VNF instance still in INSTANTIATED state and external connectivity of the VNF is changed
Terminate VNF	VNF instance in INSTANTIATED state	terminate	TerminateVnfRequest	VNF instance in NOT_INSTANTIATED state

5.3.4 Flow of automatic invocation of VNF scaling and VNF healing

This clause describes the sequence for the automatic invocation of "Scale VNF", "Scale VNF to Level" and "Heal VNF" operations by the VNFM, also known as "auto-scale" and "auto-heal". The criteria based on which the VNFM decides when to invoke an automatic scaling or automatic healing are outside the scope of the present document and can include certain changes in monitoring parameters that are monitored by the VNFM by PM jobs or thresholds, changes in VNF indicator values that are polled by the VNFM or that are reported to the VNFM via "VnfIndicatorValueChangeNotification" messages. Auto-scaling and auto-healing can be enabled and disabled by the EM by modifying the appropriate "isAutoscaleEnabled" and "isAutohealEnabled" configurable properties of the VNF using the sequence flow according to clause 5.3.6.

This flow is applicable to the automatic invocation of the following operations:

- Scale VNF

- Scale VNF to Level

- Heal VNF

Figure 5.3.4-1 illustrates the flow.
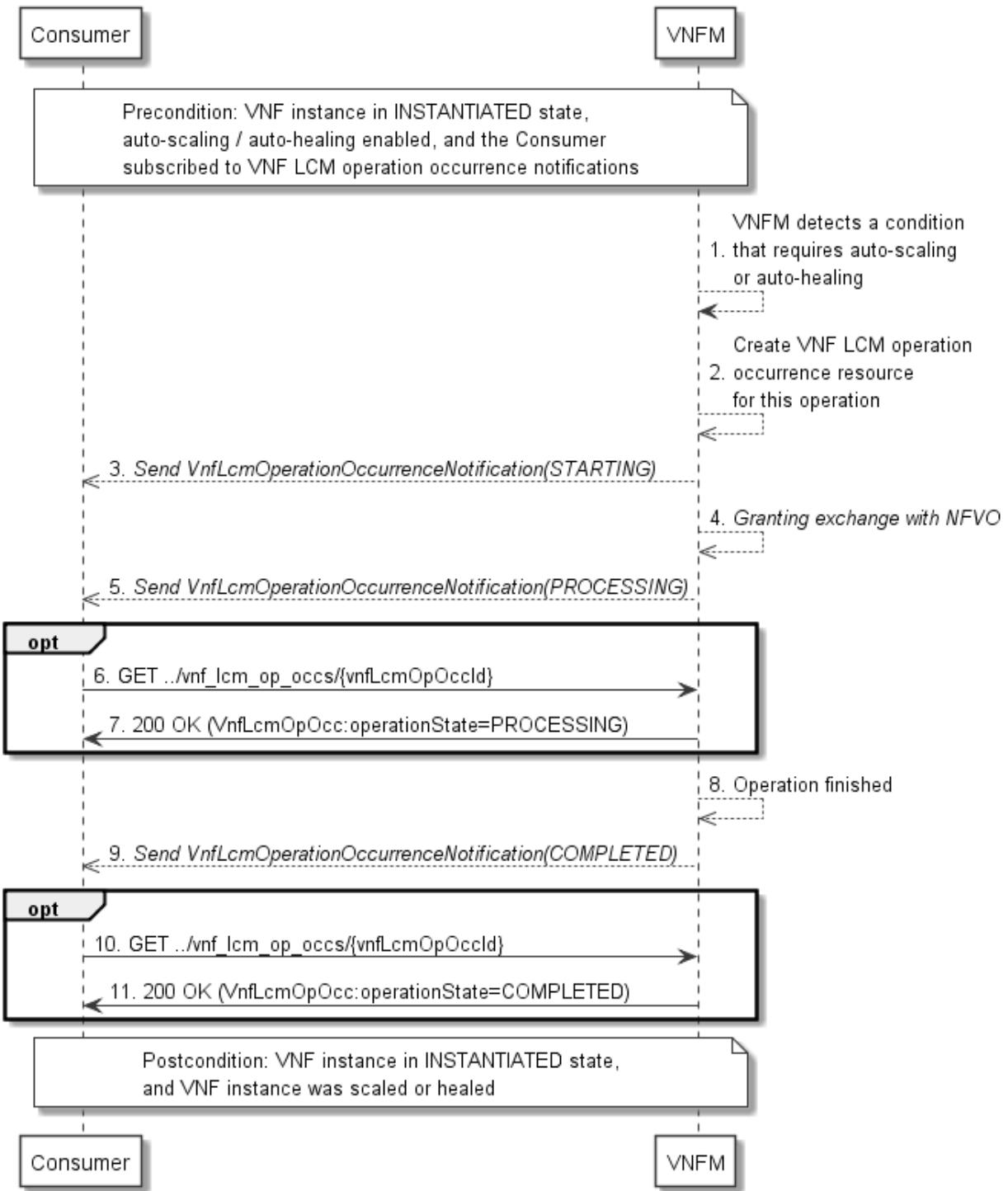
Figure 5.3.4-1: Flow of auto-scaling and auto-healing

Precondition: The VNF instance is in INSTANTIATED state, auto-scaling/auto-healing is enabled, and the Consumer is subscribed to VNF LCM operation occurrence notifications.

The automatic invocation of a VNF scaling or VNF healing operation, as illustrated in figure 5.3.4-1, consists of the following steps:

1. The VNFM detects a condition that triggers auto-scaling (Scale VNF or Scale VNF To Level) or auto-healing (Heal VNF) of the VNF, and selects the appropriate parameters for the operation.

2. The VNFM creates a "VNF LCM operation occurrence" resource for the operation.

3. The VNFM sends to the Consumer a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the start of the lifecycle management operation occurrence.

4. The VNFM and the NFVO exchange granting information.

5. The VNFM sends to the Consumer a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state.

6. If desired, the Consumer can poll the "VNF lifecycle management operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF lifecycle management operation occurrence.

7. In the response to that request, the VNFM returns to the Consumer information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".

8. The VNFM has finished the operation.

9. The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the completion of the lifecycle management operation occurrence.

10. If desired, the Consumer can send a new GET request to the "VNF lifecycle management operation occurrence" resource.

11. In the response to that request, the VNFM returns to the Consumer information about the result of the operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: The VNF instance in INSTANTIATED state, and the VNF instance was scaled or healed as appropriate.

Error handling: If the VNF lifecycle management operation fails, error information is provided in the notification message that reports the erroneous completion of the procedure, and is also available in the resource that represents the actual VNF lifecycle management operation occurrence related to this VNF lifecycle management operation.

5.3.5   Flow of the Query VNF operation

This clause describes a sequence for querying/reading information about a VNF instance.
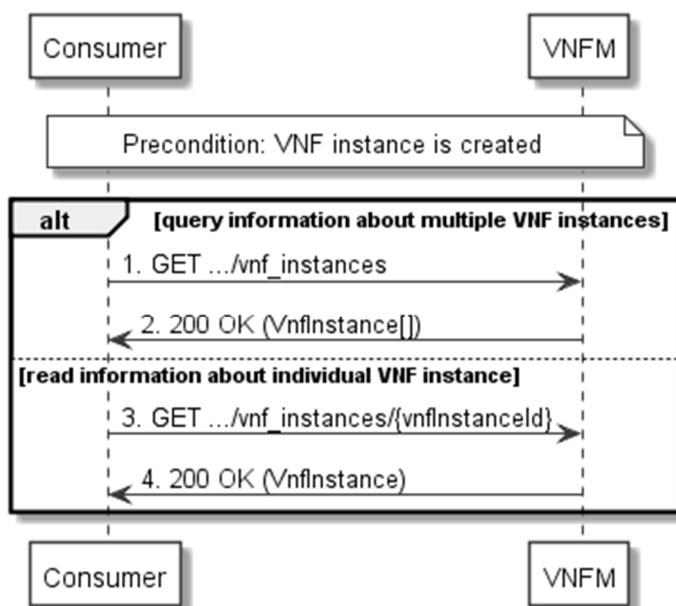
Figure 5.3.5-1: Flow of VNF instance query/read

Precondition: The resource representing the VNF instance has been created.

VNF instance query, as illustrated in figure 5.3.5-1, consists of the following steps:

1.   If the Consumer intends to query all VNF instances, it sends a GET request to the "VNF instances" resource.

2.   The VNFM returns a "200 OK" response to the Consumer, and includes zero or more data structures of type "VnfInstance" in the payload body.

3.   If the Consumer intends to read information about a particular VNF instance, it sends a GET request to the "Individual VNF instance" resource, addressed by the appropriate VNF instance identifier in its resource URI.

4.   The VNFM returns a "200 OK" response to the Consumer, and includes one data structure of type "VnfInstance" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

5.3.6   Flow of the Modify VNF Information operation

This clause describes a sequence for updating information about a VNF instance.
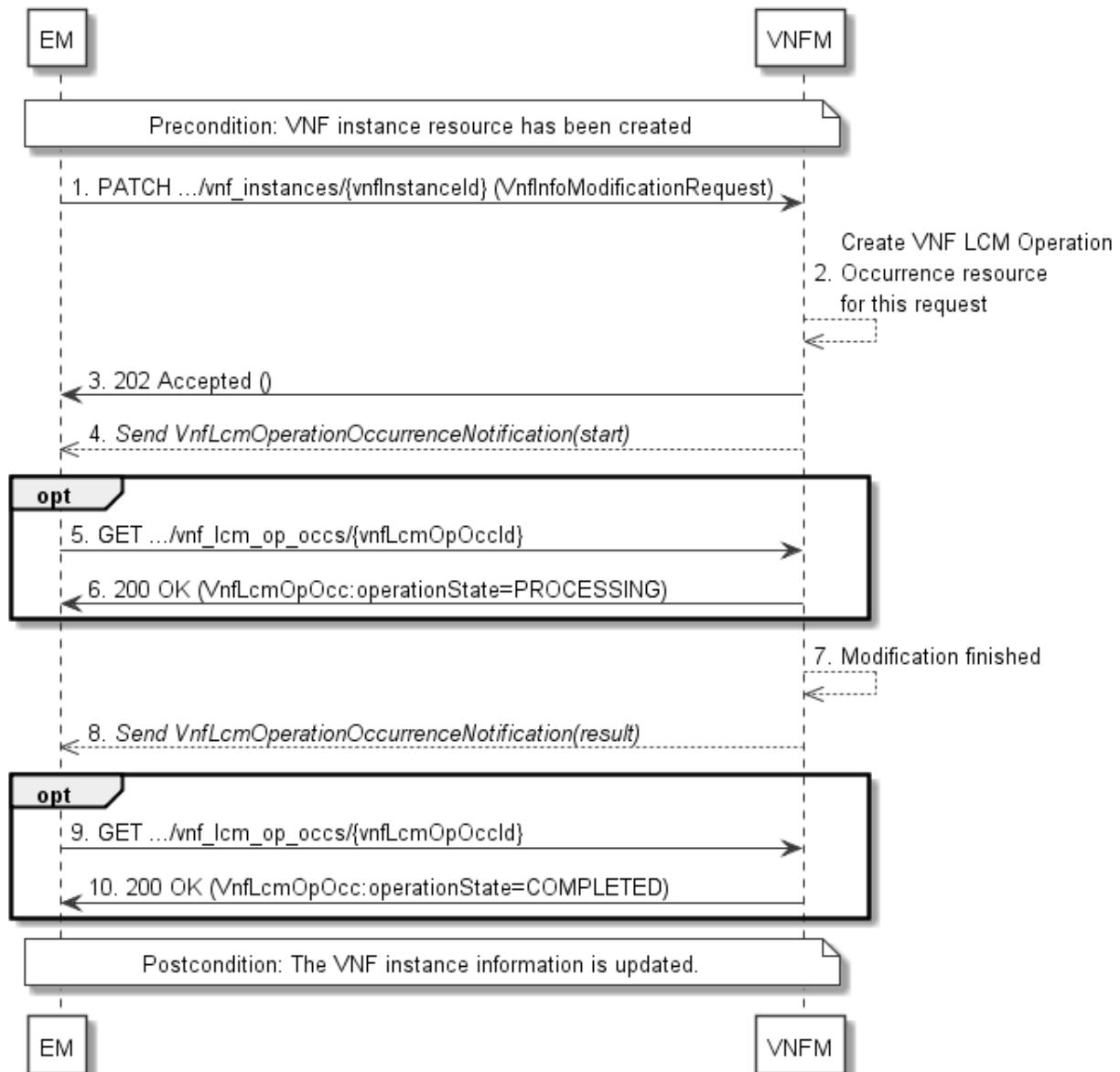
Figure 5.3.6-1: Flow of the modification of VNF instance information

NOTE: Due to possible race conditions, the 202 response and the VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The resource representing the VNF instance has been created.

Updating the VNF instance information, as illustrated in figure 5.3.6-1, consists of the following steps:

1. The EM sends a PATCH request to the "Individual VNF instance" resource of the VNF instance that is to be operated and includes in the payload body a data structure of type "VnfInfoModificationRequest".

2. The VNFM creates a "VNF LCM operation occurrence" resource for the request.

3. The VNFM returns a "202 Accepted" response with an empty payload body and a "Location" HTTP header that points to the new "VNF LCM operation occurrence" resource, i.e. it includes the URI of that resource which is ".../vnf_lcm_op_occs/{vnfLcmOpOccId}".

4. The VNFM sends to the EM a lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the start of the operation.

5. If desired, the EM can poll the "VNF LCM operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF lifecycle management operation occurrence.

6. In the response to that request, the VNFM returns to the EM information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc".

7. The VNFM has finished the modification operation.

8. The VNFM sends a lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the completion of the operation, and the performed changes.

9. If desired, the EM can send a new GET request to the "VNF LCM operation occurrence" resource.

10. In the response to that request, the VNFM returns to the EM information about the result of the operation, by providing in the payload body a data structure of type "VnfLcmOpOcc".

Postcondition: Upon successful completion, information of the VNF instance is updated.

Error handling: If the updating of VNF instance information fails, error information is provided in the notification message that reports the erroneous completion of the procedure, and is also available in the resource that represents the actual VNF lifecycle management operation occurrence related to this VNF LCM operation.

5.3.7   Flow of the Get Operation Status operation

This clause describes a sequence for obtaining the status of a VNF lifecycle management operation occurrence.
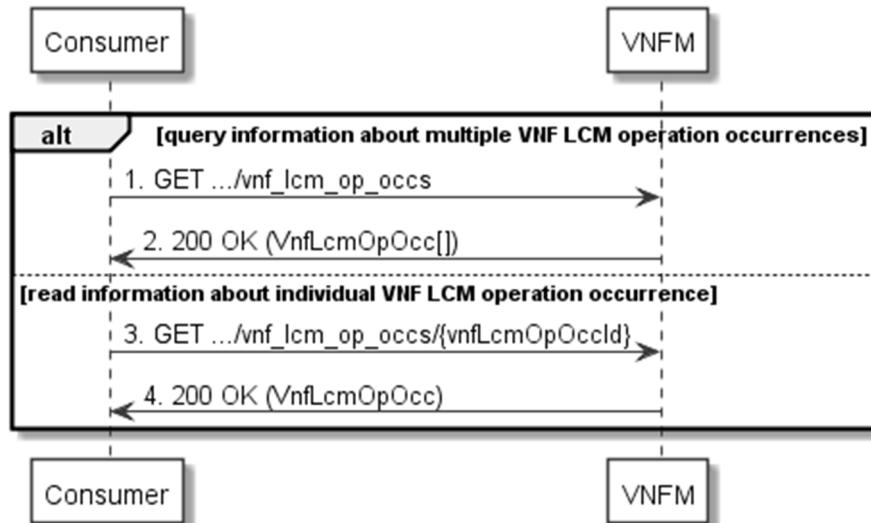
Figure 5.3.7-1: Flow of Get VNF lifecycle operation status

Obtaining the VNF lifecycle operation status, as illustrated in figure 5.3.7-1, consists of the following steps:

1)   If the Consumer intends to query all VNF lifecycle management operation occurrences, it sends a GET request to the "VNF LCM operation occurrences" resource.

2)   The VNFM returns a "200 OK" response to the Consumer, and includes zero or more data structures of type "VnfLcmOpOcc" in the payload body.

3)   If the Consumer intends to read information about a particular VNF LCM operation occurrence, it sends a GET request to the "Individual VNF LCM operation occurrence" resource, addressed by the appropriate VNF LCM operation occurrence identifier in its resource URI.

4)   The VNFM returns a "200 OK" response to the Consumer, and includes one data structure of type "VnfLcmOpOcc" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

5.3.8   Flow of managing subscriptions

This clause describes the procedure for creating, querying/reading and terminating subscriptions to notifications related to VNF lifecycle management.
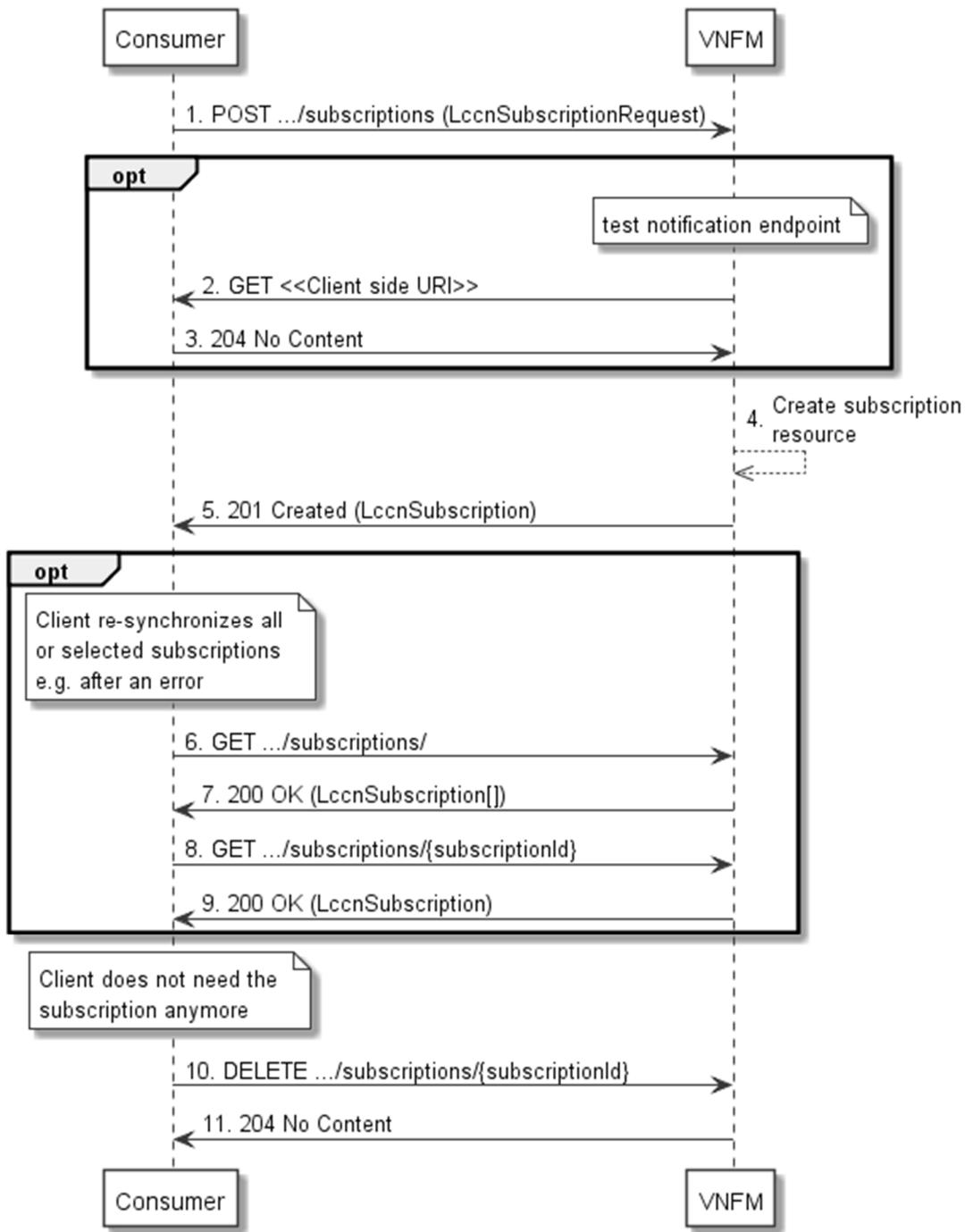
Figure 5.3.8-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 5.3.8-1.

1)   The Consumer sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "LccnSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2)   Optionally, to test the notification endpoint that was registered by the Consumer as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3)   In that case, the Consumer returns a "204 No Content" response to indicate success.

4)   The VNFM creates a new subscription to notifications related to VNF lifecycle changes, and a resource that represents this subscription.

5)   The VNFM returns a 201 Created response containing a data structure of type "LccnSubscription" representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6)   If desired, e.g. to recover from an error situation, the Consumer may query information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7)   In that case, the VNFM returns a "200 OK" response that contains zero or more representations of all existing subscriptions that were created by the Consumer.

8)   If desired, e.g. to recover from an error situation, the Consumer may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9)   In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10)   If the Consumer does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11)   The Consumer acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is nor valid: endpoint cannot be reached, subscription information is malformed, etc.

5.3.9   Flow of sending notifications

This clause describes the procedure for sending notifications.
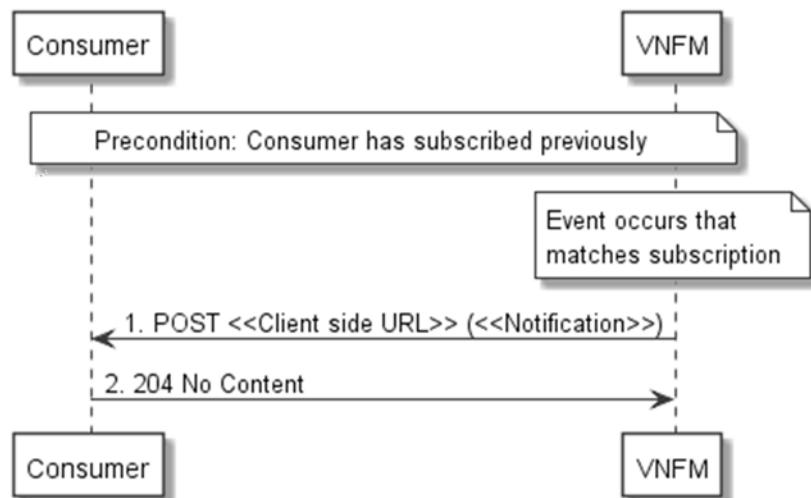
Figure 5.3.9-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in figure 5.3.9-1.

Precondition: The Consumer has subscribed previously to notifications related to VNF lifecycle management:

1)   If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the Consumer has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API (see clauses 7.5.2.5, 7.5.2.6 and 7.5.2.8).

2)   The Consumer acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the Consumer, it can retry sending the notification.

5.3.10   Flow of retrying a VNF lifecycle management operation

This clause describes a sequence for retrying a VNF lifecycle management operation occurrence that is represented by a VNF LCM operation occurrence resource. Retry is used if an operation is in FAILED_TEMP state, and there is reason to believe that the operation will eventually succeed when retried, for instance because obstacle that led to an error during the execution of the LCM operation have been removed by an automated procedure, or by manual intervention. The "retry" operation is also called "idempotent retry" because it is possible to invoke retry multiple times, without side effects.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.
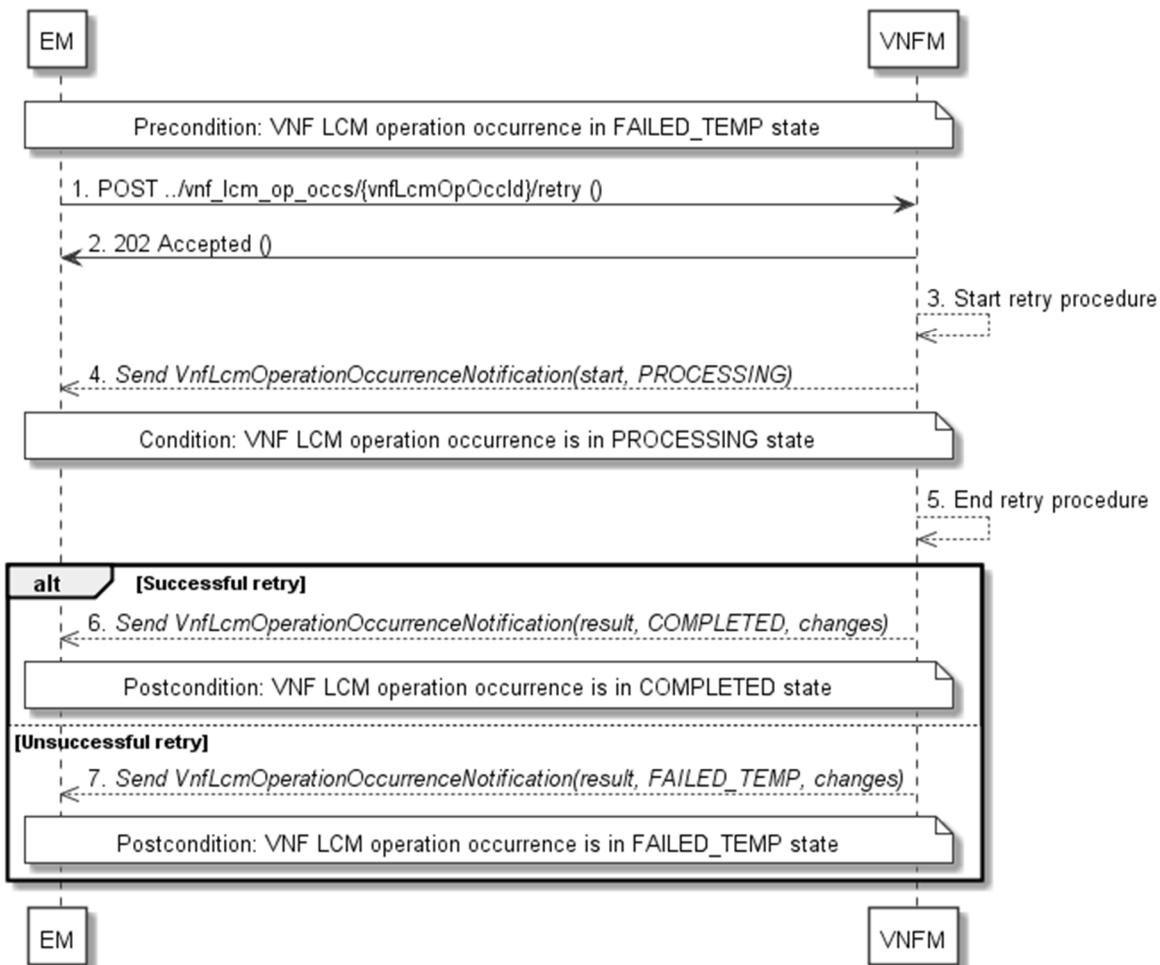
Figure 5.3.10-1: Flow of retrying a VNF lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "PROCESSING" VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Retrying a VNF lifecycle operation, as illustrated in figure 5.3.10-1, consists of the following steps:

1)   The EM sends a POST request with an empty body to the "Retry operation task" resource of the VNF LCM operation occurrence that is to be retried.

2)   The VNFM returns a "202 Accepted" response.

3)   The VNFM starts the retry procedure.

4)   The VNFM sends a VNF lifecycle management operation occurrence notification of type "start" to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state.

5)   The VNFM finishes the retry procedure.

6)   On successful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate successful completion of the operation, and inform the EM about the virtualised resources changes.

7)   On unsuccessful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (retry failed) of the operation, and to inform the EM about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in one of the following states: FAILED_TEMP, COMPLETED. COMPLETED is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the VNF LCM operation occurrence resource is in any other state than FAILED_TEMP, or in case Retry is not supported by for the particular VNF LCM operation for the particular VNF.

5.3.11   Flow of rolling back a VNF lifecycle management operation

This clause describes a sequence for rolling back a VNF lifecycle management operation occurrence that is represented by a VNF LCM operation occurrence resource. Rollback can be used for example if an operation is in FAILED_TEMP state, and there is no reason to believe that retrying the operation will eventually succeed.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.
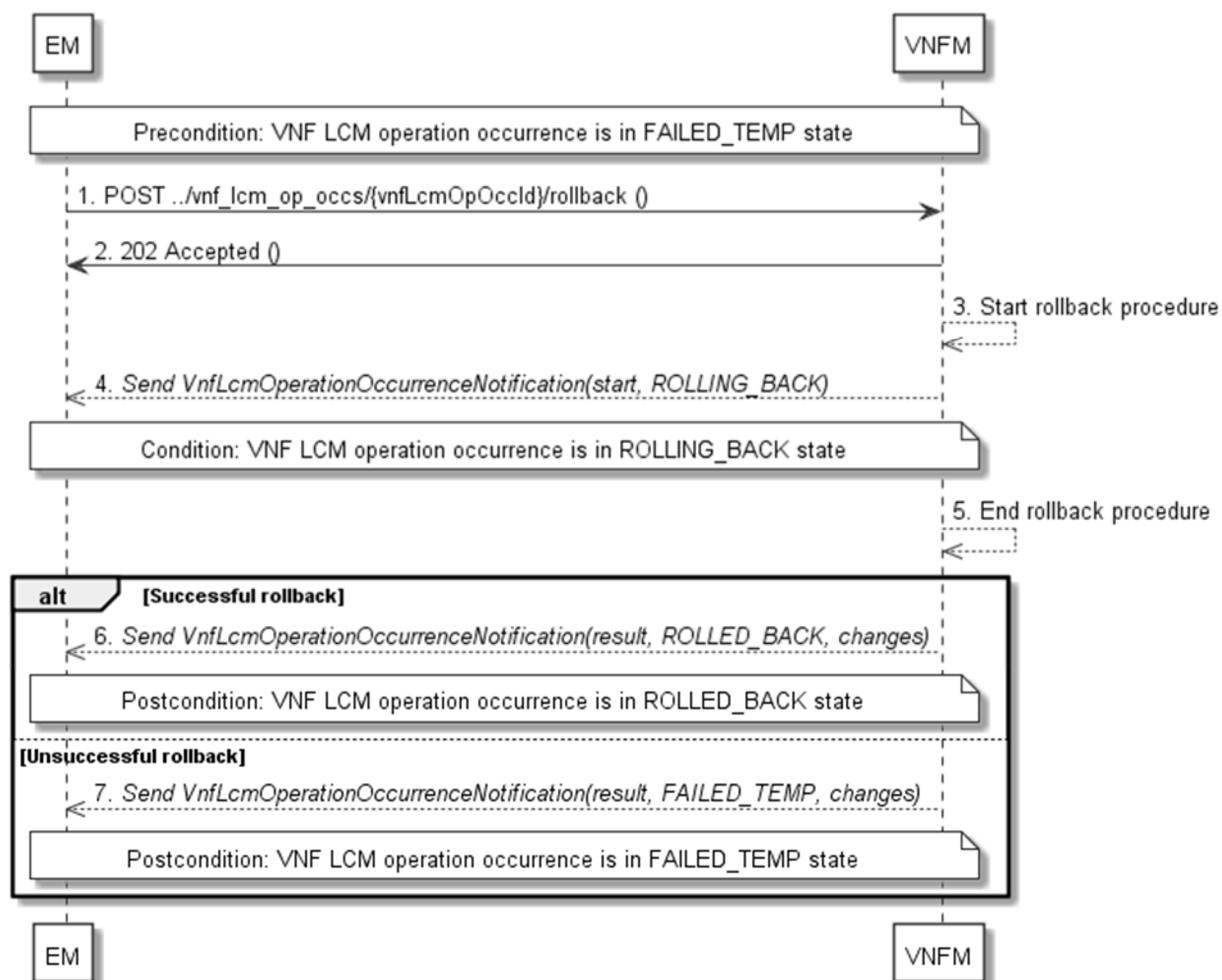
Figure 5.3.11-1: Flow of rolling back a VNF lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "ROLLING_BACK" VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Initiating the rollback of a VNF lifecycle management operation, as illustrated in figure 5.3.11-1, consists of the following steps:

1) The EM sends a POST request with an empty body to the "Rollback operation task" resource of the VNF LCM operation occurrence that is to be rolled back.

2) The VNFM returns a "202 Accepted" response.

3) The VNFM starts the rollback procedure.

4) The VNFM sends a VNF lifecycle management operation occurrence notification of type "start" to indicate that the VNF LCM operation occurrence enters the "ROLLING_BACK" state.

5) The VNFM finishes the rollback procedure.

6) On successful rollback, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate successful completion of the operation, and inform the EM about the virtualised resources changes.

7) On unsuccessful retry, the VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (rollback failed) of the operation, and to inform the EM about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in one of the following states: FAILED_TEMP, ROLLED_BACK. ROLLED_BACK is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than FAILED_TEMP, or in case Rollback is not supported for the particular VNF LCM operation for the particular VNF.

5.3.12   Flow of failing a VNF lifecycle management operation

This clause describes a sequence for declaring as "failed" a VNF lifecycle management operation occurrence that is represented by a VNF LCM operation occurrence resource. If there is neither an assumption that the operation can eventually succeed after further retries, nor that the operation can be successfully rolled back, the operation can be declared as "failed". This will unblock the invocation of other LCM operations, such as HealVnf, or non-graceful VNF termination, on the affected VNF instance.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.
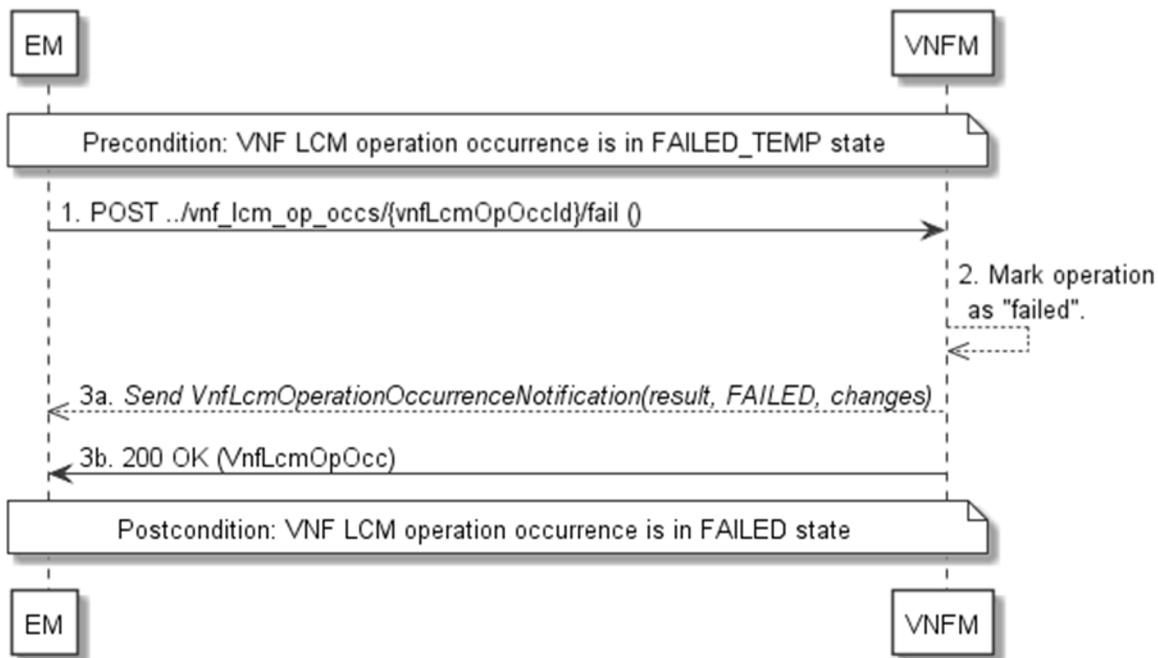
Figure 5.3.12-1: Flow of declaring a VNF lifecycle management operation as failed

NOTE: Due to possible race conditions, the 200 response and the "FAILED"
VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The VNF lifecycle management operation occurrence is in FAILED_TEMP state.

Declaring a VNF lifecycle management operation as failed, as illustrated in figure 5.3.12-1, consists of the following steps:

1) The EM sends a POST request with an empty body to the "Fail operation task" resource of the VNF LCM operation occurrence that is to be marked as failed.

2) The VNFM marks the operation as failed.

3) The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate the final failure of the operation, and to inform the EM about the virtualised resources changes.

4) Furthermore, it returns a "200 OK" response, and includes in the body a VnfLcmOpOcc structure. The order in which the response and the notification arrive at the EM is not defined.

Postcondition: The VNF lifecycle management operation occurrence is FAILED state. This is a terminal state (see clause 5.6.2.2).

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than FAILED_TEMP.

5.3.13 Flow of cancelling a VNF lifecycle management operation

This clause describes a sequence for cancelling an ongoing VNF LCM operation occurrence, or a rollback of a VNF LCM operation occurrence. The possibility and timing of cancellation is dependent on the implementation of the underlying lifecycle management operation.

A comprehensive description of the handling of VNF lifecycle management errors is provided in clause 5.6.
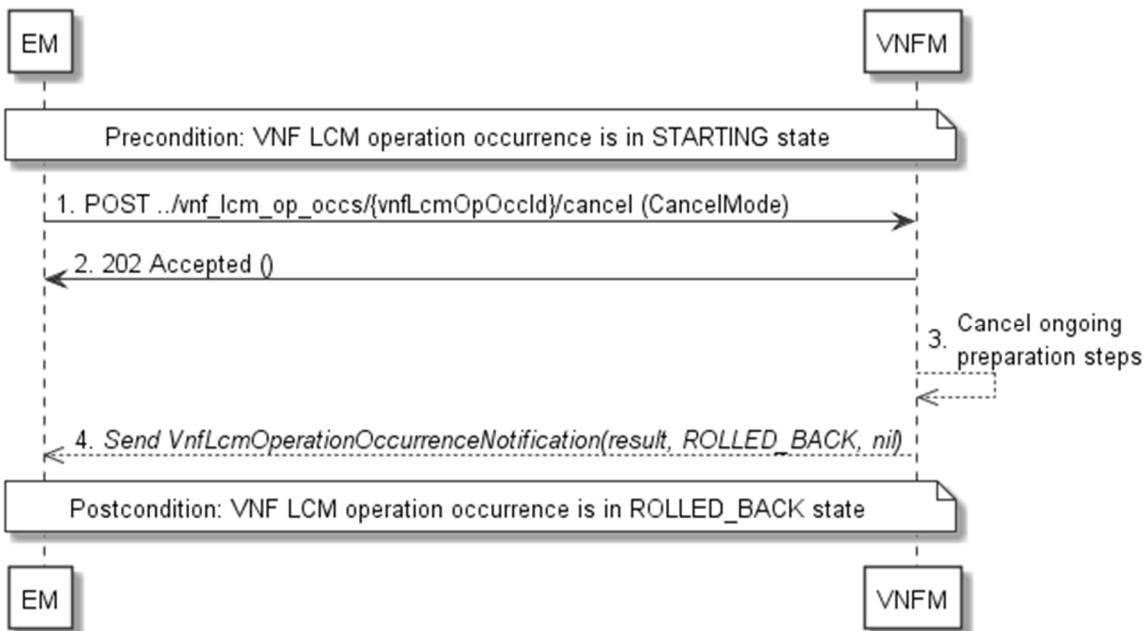
Figure 5.3.13-1: Flow of cancelling a VNF lifecycle management operation in "STARTING" state

NOTE: Due to possible race conditions, the 202 response and the "ROLLED_BACK"
         VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The VNF lifecycle management operation occurrence is in STARTING state.

Cancelling a VNF lifecycle operation when it is in STARTING state, as illustrated in figure 5.3.13-1, consists of the following steps:

1)   The EM sends a POST request with a "CancelMode" structure in the body to the "Cancel operation task" resource of the VNF LCM operation occurrence that is to be cancelled.

2)   The VNFM returns a "202 Accepted" response.

3)   The VNFM cancels the ongoing preparation operations.

4)   The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (cancelled) of the operation, and to inform the EM that there were no virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is in ROLLED_BACK state.

Error handling: The operation is rejected in case the VNF lifecycle operation occurrence is in any other state than STARTING.
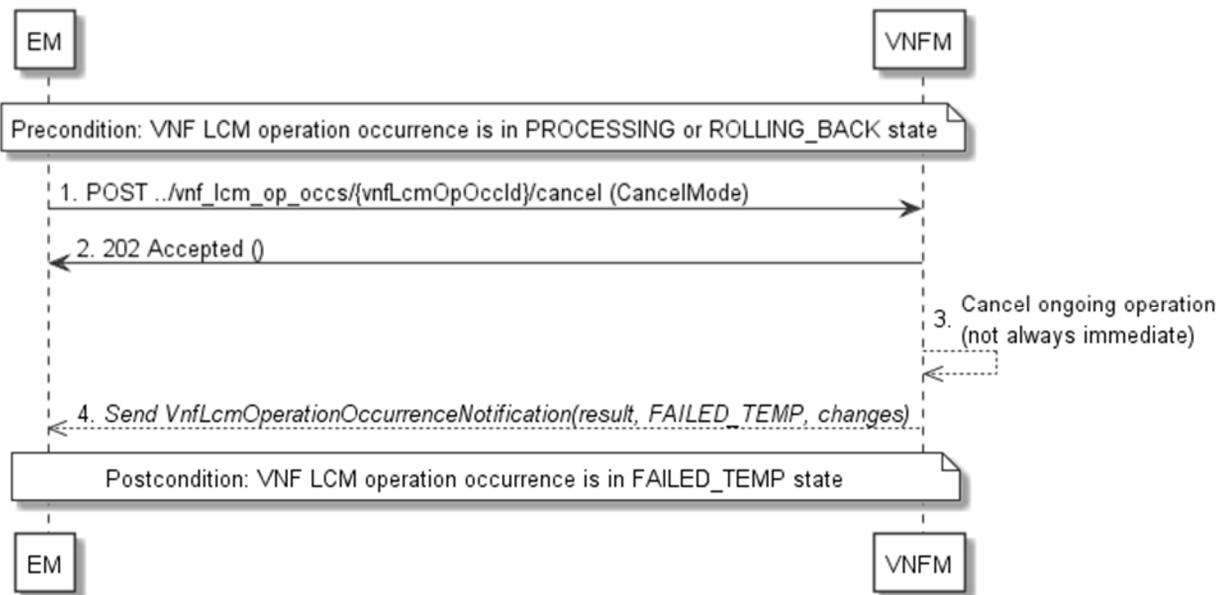
**Figure 5.3.13-2: Flow of cancelling a VNF lifecycle management operation
in "PROCESSING" or "ROLLING_BACK" state**

NOTE: Due to possible race conditions, the 202 response and the "FAILED_TEMP"
           VnfLcmOperationOccurrenceNotification can arrive in any order at the EM.

Precondition: The VNF lifecycle management operation occurrence is in PROCESSING or ROLLING_BACK state.

Cancelling a VNF lifecycle operation when it is in "PROCESSING" or "ROLLING_BACK" state, as illustrated in figure 5.3.13-2, consists of the following steps:

1)   The EM sends a POST request with a "CancelMode" structure in the body to the "Cancel operation task" resource of the VNF LCM operation occurrence that is to be cancelled.

2)   The VNFM returns a "202 Accepted" response.

3)   The VNFM cancels the ongoing LCM operation. This can take some time.

4)   The VNFM sends a VNF lifecycle management operation occurrence notification (see clause 5.3.9) to indicate an intermediate error (cancelled) of the operation, and to inform the EM about the virtualised resources changes.

Postcondition: The VNF lifecycle management operation occurrence is FAILED_TEMP state.

Error handling: The operation is rejected in case the VNF lifecycle management operation occurrence is in any other state than PROCESSING or ROLLING_BACK, or in case Cancel is not supported for the particular VNF LCM operation for the particular VNF.

5.4   Resources

5.4.1   Introduction

This clause defines all the resources and methods provided by the VNF lifecycle management interface.

5.4.2   Resource: VNF instances

5.4.2.1   Description

This resource represents VNF instances. The client can use this resource to create individual VNF instance resources, and to query VNF instances.

5.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances

This resource shall support the resource URI variables defined in table 5.4.2.2-1.

Table 5.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.2.3 Resource methods

5.4.2.3.1 POST

The POST method creates a new VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.2.3.1-1 and 5.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.2.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	CreateVnfRequest	1		The VNF creation parameters, as defined in clause 5.5.2.3
Response body	Data type	Cardinality	Response Codes	Description
	VnfInstance	1	201 Created	A VNF Instance identifier was created successfully. The response body shall contain a representation of the created VNF instance, as defined in clause 5.5.2.2. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created VNF instance.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.2.3.2 GET

The GET method queries information about multiple VNF instances.

This method shall follow the provisions specified in the tables 5.4.2.3.2-1 and 5.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. All attribute names that appear in the VnfInstance and in data types referenced from it shall be supported in attribute-based filtering parameters.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the VnfInstance structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - vnfConfigurableProperties - instantiatedVnfInfo - metadata - extensions

Table 5.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**	
	VnfInstance	0..N	200 OK	Information about zero or more VNF instances was queried successfully. The response body shall contain representations of zero or more VNF instances, as defined in clause 5.5.2.2.	
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.	
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

5.4.2.3.3   PUT

Not supported.

5.4.2.3.4   PATCH

Not supported.

5.4.2.3.5   DELETE

Not supported.

5.4.3   Resource: Individual VNF instance

5.4.3.1   Description

This resource represents an individual VNF instance. The client can use this resource to modify and delete the underlying VNF instance, and to read information about the VNF instance.

5.4.3.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}

The base resource URI variables for this resource are defined in table 5.4.3.2-1.

Table 5.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.3.3   Resource methods

5.4.3.3.1   POST

Not supported.

5.4.3.3.2   GET

The GET method retrieves information about a VNF instance by reading an individual VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.3.3.2-1 and 5.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
	Data type	**Cardinality**	**Response codes**	**Description**
Response body	VnfInstance	1	200 OK	Information about an individual VNF instance was queried successfully. The response body shall contain a representation of the VNF instance, as defined in clause 5.5.2.2.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.3.3.3   PUT

Not supported.

5.4.3.3.4   PATCH

This method modifies an individual VNF instance resource.

Changes to the VNF configurable properties are applied to the configuration in the VNF instance, and are reflected in the representation of this resource. Other changes are applied to the VNF instance information managed by the VNFM, and are reflected in the representation of this resource.

This method shall follow the provisions specified in the tables 5.4.3.3.4-1 and 5.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality	Description	
Request body	VnfInfoModificationRequest	1	Parameters for the VNF modification, as defined in clause 5.5.2.12. The Content-Type header shall be set to "application/merge-patch+json" according to IETF RFC 7396 [15].	

	Data type	Cardinality	Response Codes	Description
Response body	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. On success, the HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation. The response body shall be empty.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF instance resource. Typically, this is due to the fact that another LCM operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	412 Precondition failed	Error: A precondition given in an HTTP request header is not fulfilled. Typically, this is due to an ETag mismatch, indicating that the resource was modified by another entity. The response body should contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.3.3.5   DELETE

This method deletes an individual VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.3.3.5-1 and 5.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.3.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The VNF instance resource and the associated VNF identifier were deleted successfully. The response body shall be empty.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in INSTANTIATED state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.4   Resource: Instantiate VNF task

5.4.4.1   Description

This task resource represents the "Instantiate VNF" operation. The client can use this resource to instantiate a VNF instance.

5.4.4.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/instantiate

This resource shall support the resource URI variables defined in table 5.4.4.2-1.

Table 5.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance to be instantiated. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.

5.4.4.3   Resource methods

5.4.4.3.1   POST

The POST method instantiates a VNF instance.

This method shall follow the provisions specified in the tables 5.4.4.3.1-1 and 5.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.4.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	InstantiateVnfRequest	1		Parameters for the VNF instantiation, as defined in clause 5.5.2.4.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in INSTANTIATED state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.4.3.2   GET

Not supported.

5.4.4.3.3   PUT

Not supported.

5.4.4.3.4   PATCH

Not supported.

5.4.4.3.5   DELETE

Not supported.

5.4.5   Resource: Scale VNF task

5.4.5.1   Description

This task resource represents the "Scale VNF" operation. The client can use this resource to request scaling a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

See clause B.2 in Annex B for an explanation of VNF scaling.

5.4.5.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/scale

This resource shall support the resource URI variables defined in table 5.4.5.2-1.

Table 5.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be scaled. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.

5.4.5.3   Resource methods

5.4.5.3.1   POST

The POST method requests to scale a VNF instance resource incrementally.

This method shall follow the provisions specified in the tables 5.4.5.3.1-1 and 5.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.5.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	ScaleVnfRequest	1		Parameters for the scale VNF operation, as defined in clause 5.5.2.5.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.5.3.2   GET

Not supported.

5.4.5.3.3   PUT

Not supported.

5.4.5.3.4   PATCH

Not supported.

5.4.5.3.5   DELETE

Not supported.

5.4.6 Resource: Scale VNF to Level task

5.4.6.1 Description

This task resource represents the "Scale VNF to Level" operation. The client can use this resource to request scaling of a VNF instance to a target level.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

See clause B.2 for an explanation of VNF scaling.

5.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/scale_to_level

This resource shall support the resource URI variables defined in table 5.4.6.2-1.

Table 5.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be scaled to a target level. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.

5.4.6.3 Resource methods

5.4.6.3.1 POST

The POST method requests to scale a VNF instance resource to a target level.

This method shall follow the provisions specified in the tables 5.4.6.3.1-1 and 5.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.6.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.6.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	ScaleVnfToLevelRequest	1	Parameters for the scale VNF to Level operation, as defined in clause 5.5.2.6.	

Response body	Data type	Cardinality	Response Codes	Description
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource, consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.6.3.2   GET

Not supported.

5.4.6.3.3   PUT

Not supported.

5.4.6.3.4   PATCH

Not supported.

5.4.6.3.5   DELETE

Not supported.

5.4.7   Resource: Change VNF Flavour task

5.4.7.1   Description

This task resource represents the "Change VNF Flavour" operation. The client can use this resource to change the deployment flavour for a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF. This operation may be service-disruptive.

5.4.7.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/change_flavour

This resource shall support the resource URI variables defined in table 5.4.7.2-1.

Table 5.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance of which the deployment flavour is requested to be changed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.7.3   Resource methods

5.4.7.3.1   POST

The POST method changes the deployment flavour of a VNF instance.

This method shall follow the provisions specified in the tables 5.4.7.3.1-1 and 5.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	ChangeVnfFlavourRequest	1		Parameters for the Change VNF Flavour operation, as defined in clause 5.5.2.7.

Response body	Data type	Cardinality	Response Codes	Description
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.7.3.2   GET

Not supported.

5.4.7.3.3   PUT

Not supported.

5.4.7.3.4   PATCH

Not supported.

5.4.7.3.5   DELETE

Not supported.

5.4.8 Resource: Terminate VNF task

5.4.8.1 Description

This task resource represents the "Terminate VNF" operation. The client can use this resource to terminate a VNF instance.

5.4.8.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/terminate

This resource shall support the resource URI variables defined in table 5.4.8.2-1.

Table 5.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	The identifier of the VNF instance to be terminated. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.		

5.4.8.3 Resource methods

5.4.8.3.1 POST

The POST method terminates a VNF instance.

This method shall follow the provisions specified in the tables 5.4.8.3.1-1 and 5.4.8.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.8.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.8.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	TerminateVnfRequest	1		Parameters for the VNF termination, as defined in clause 5.5.2.8.
Response body	Data type	Cardinality	Response Codes	Description
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.8.3.2 GET

Not supported.

5.4.8.3.3 PUT

Not supported.

5.4.8.3.4 PATCH

Not supported.

5.4.8.3.5 DELETE

Not supported.

5.4.9 Resource: Heal VNF task

5.4.9.1 Description

This task resource represents the "Heal VNF" operation. The client can use this resource to request healing a VNF instance.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

5.4.9.2 Resource definition

The resource URI is:

   {apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/heal

This resource shall support the resource URI variables defined in table 5.4.9.2-1.

Table 5.4.9.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be healed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.9.3   Resource methods

5.4.9.3.1   POST

The POST method requests to heal a VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.9.3.1-1 and 5.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	HealVnfRequest	1		Parameters for the Heal VNF operation, as defined in clause 5.5.2.9.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be that the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.9.3.2   GET

Not supported.

5.4.9.3.3   PUT

Not supported.

5.4.9.3.4   PATCH

Not supported.

5.4.9.3.5   DELETE

Not supported.

5.4.10   Resource: Operate VNF task

5.4.10.1   Description

This task resource represents the "Operate VNF" operation. The client can use this resource to operate a VNF.

It depends on the VNF capabilities, and is declared in the VNFD, whether this operation is supported for a particular VNF.

The "Operate VNF" operation enables requesting to change the operational state of a VNF instance, including starting and stopping the VNF instance.

  NOTE 1: These operations are complementary to instantiating and terminating a VNF.

  NOTE 2: In the present document, only starting and stopping the VNF instances are supported. Extension of this operation to support other VNF state changes is left for future specification.

A VNF instance can be in the following states:

  • STARTED: the VNF instance is up and running.

  • STOPPED: the VNF instance has been shut down. A VNF instance is stopped if all its VNFC instances are also stopped.

In the state STOPPED, the virtualisation containers, where the VNFC instances of the VNF run, are shut down but not deleted. In addition, if the workflow requires a graceful stop, as part of this process the VNFM (producer of the interface) will interact with VNF/EM to gracefully stop the VNF application. Once a VNF is instantiated, i.e. all instantiation steps have been completed, the VNF instance is in the state STARTED.

5.4.10.2   Resource definition

The resource URI is:

  {apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/operate

This resource shall support the resource URI variables defined in table 5.4.10.2-1.

Table 5.4.10.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to be operated. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.10.3   Resource methods

5.4.10.3.1   POST

The POST method changes the operational state of a VNF instance resource.

This method shall follow the provisions specified in the tables 5.4.10.3.1-1 and 5.4.10.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.10.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.10.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	OperateVnfRequest	1		Parameters for the Operate VNF operation, as defined in clause 5.5.2.10.

	Data type	Cardinality	Response Codes	Description
Response body	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the operation.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF instance represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that the VNF instance resource is in NOT_INSTANTIATED state, or that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.10.3.2   GET

Not supported.

5.4.10.3.3   PUT

Not supported.

5.4.10.3.4   PATCH

Not supported.

5.4.10.3.5   DELETE

Not supported.

5.4.11   Resource: Change external VNF connectivity task

5.4.11.1   Description

This task resource represents the "Change external VNF connectivity" operation. The client can use this resource to change the external connectivity of a VNF instance. The types of changes that this operation supports are:

-    Disconnect the external CPs that are connected to a particular external VL, and connect them to a different external VL.

-    Change the connectivity parameters of the existing external CPs, including changing addresses.

NOTE: Depending on the capabilities of the underlying VIM resources, certain changes (e.g. modifying the IP address assignment) might not be supported without deleting the resource and creating another one with the modified configuration.

VNFs shall support this operation. This operation may be service-disruptive.

5.4.11.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_instances/{vnfInstanceId}/change_ext_conn

This resource shall support the resource URI variables defined in table 5.4.11.2-1.

Table 5.4.11.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance of which the external connectivity is requested to be changed. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.11.3   Resource methods

5.4.11.3.1   POST

The POST method changes the external connectivity of a VNF instance.

This method shall follow the provisions specified in the tables 5.4.11.3.1-1 and 5.4.11.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.11.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.11.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description	
	ChangeExtVnfConnectivityRequest	1		Parameters for the Change external VNF connectivity operation, as defined in clause 5.5.2.11.	
Response body	Data type	Cardinality	Response Codes	Description	
	n/a		202 Accepted	The request was accepted for processing, but the processing has not been completed. The response body shall be empty. The HTTP response shall include a "Location" HTTP header that contains the URI of the newly-created "VNF LCM operation occurrence" resource corresponding to the instantiation operation.	
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is due to the fact that another lifecycle management operation is ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

5.4.11.3.2   GET

Not supported.

5.4.11.3.3   PUT

Not supported.

5.4.11.3.4   PATCH

Not supported.

5.4.11.3.5   DELETE

Not supported.

5.4.12   Resource: VNF LCM operation occurrences

5.4.12.1   Description

This resource represents VNF lifecycle management operation occurrences. The client can use this resource to query status information about multiple VNF lifecycle management operation occurrences.

5.4.12.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs

The base resource URI variables for this resource are defined in table 5.4.12.2-1.

ETSI

Table 5.4.12.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.12.3   Resource methods

5.4.12.3.1   POST

Not supported.

5.4.12.3.2   GET

The client can use this method to query status information about multiple VNF lifecycle management operation occurrences.

This method shall follow the provisions specified in the tables 5.4.12.3.2-1 and 5.4.12.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.12.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM/VNF may supply filtering parameters. All attribute names that appear in the VnfLcmOpOcc and in data types referenced from it shall be supported in attribute-based filtering parameters.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the VnfLcmOpOcc structure in the response body if this parameter is provided, or none of the parameters "all_fields", "fields", "exclude_fields", "exclude_default" are provided: - operationParams - error - resourceChanges - changedInfo - changedExtConnectivity

Table 5.4.12.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VnfLcmOpOcc	0..N	200 OK	Status information for zero or more VNF lifecycle management operation occurrences was queried successfully. The response body shall contain status information about zero or more VNF lifecycle operation occurrences, as defined in clause 5.5.2.13.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.12.3.3   PUT

Not supported.

5.4.12.3.4   PATCH

Not supported.

5.4.12.3.5   DELETE

Not supported.

5.4.13   Resource: Individual VNF LCM operation occurrence

5.4.13.1   Description

This resource represents a VNF lifecycle management operation occurrence. The client can use this resource to read status information about an individual VNF lifecycle management operation occurrence. Further, the client can use task resources which are children of this resource to request cancellation of an operation in progress, and to request the handling of operation errors via retrying the operation, rolling back the operation, or permanently failing the operation.

5.4.13.2   Resource definition

The resource URI is:

   {apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}

The base resource URI variables for this resource are defined in table 5.4.13.2-1.

Table 5.4.13.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.

5.4.13.3   Resource methods

5.4.13.3.1   POST

Not supported.

5.4.13.3.2   GET

The client can use this method to retrieve status information about a VNF lifecycle management operation occurrence by reading an individual VNF LCM operation occurrence resource.

This method shall follow the provisions specified in the tables 5.4.13.3.2-1 and 5.4.13.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.13.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.13.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	VnfLcmOpOcc	1	200 OK	Information about an individual VNF instance was queried successfully. The response body shall contain status information about a VNF lifecycle management operation occurrence (see clause 5.5.2.13).	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

5.4.13.3.3   PUT

Not supported.

5.4.13.3.4   PATCH

Not supported.

5.4.13.3.5   DELETE

Not supported.

5.4.14   Resource: Retry operation task

5.4.14.1   Description

This task resource represents the "Retry operation" operation. The client can use this resource to initiate retrying a VNF lifecycle operation.

5.4.14.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/retry

This resource shall support the resource URI variables defined in table 5.4.14.2-1.

Table 5.4.14.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be retried. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.14.3   Resource methods

5.4.14.3.1   POST

The POST method initiates retrying a VNF lifecycle operation if that operation has experienced a temporary failure, i.e. the related "VNF LCM operation occurrence" resource is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.14.3.1-1 and 5.4.14.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.14.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.14.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			The POST request to this resource has an empty payload body.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state or another error handling action is starting such as rollback or fail. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.14.3.2   GET

Not supported.

5.4.14.3.3   PUT

Not supported.

5.4.14.3.4   PATCH

Not supported.

5.4.14.3.5   DELETE

Not supported.

5.4.15   Resource: Rollback operation task

5.4.15.1   Description

This task resource represents the "Rollback operation" operation. The client can use this resource to initiate rolling back a VNF lifecycle operation.

5.4.15.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/rollback

This resource shall support the resource URI variables defined in table 5.4.15.2-1.

Table 5.4.15.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be rolled back. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.15.3   Resource methods

5.4.15.3.1   POST

The POST method initiates rolling back a VNF lifecycle operation if that operation has experienced a temporary failure, i.e. the related "VNF LCM operation occurrence" resource is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.15.3.1-1 and 5.4.15.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.15.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.15.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			The POST request to this resource has an empty payload body.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state or another error handling action is starting such as retry or fail. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.15.3.2   GET

Not supported.

5.4.15.3.3   PUT

Not supported.

5.4.15.3.4   PATCH

Not supported.

5.4.15.3.5   DELETE

Not supported.

5.4.16   Resource: Fail operation task

5.4.16.1   Description

This task resource represents the "Fail operation" operation. The client can use this resource to mark a VNF lifecycle management operation occurrence as "finally failed", i.e. change the state of the related VNF LCM operation occurrence resource to "FAILED", if it is not assumed that a subsequent retry or rollback will succeed. Once the operation is marked as "finally failed", it cannot be retried or rolled back anymore.

5.4.16.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/fail

This resource shall support the resource URI variables defined in table 5.4.16.2-1.

Table 5.4.16.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be marked as "failed". See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.16.3   Resource methods

5.4.16.3.1   POST

The POST method marks a VNF lifecycle management operation occurrence as "finally failed" if that operation occurrence is in "FAILED_TEMP" state.

This method shall follow the provisions specified in the tables 5.4.16.3.1-1 and 5.4.16.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.16.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.16.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			The POST request to this resource has an empty payload body.
n/Respons e body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfLcmOpOcc		200 OK	The state of the VNF lifecycle management operation occurrence was changed successfully. The response body shall include a representation of the VNF lifecycle operation occurrence resource.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the VNF LCM operation occurrence is not in FAILED_TEMP state or another error handling action is starting such as retry or rollback. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.16.3.2   GET

Not supported.

5.4.16.3.3   PUT

Not supported.

5.4.16.3.4   PATCH

Not supported.

5.4.16.3.5   DELETE

Not supported.

5.4.17   Resource: Cancel operation task

5.4.17.1   Description

This task resource represents the "Cancel operation" operation. The client can use this resource to cancel an ongoing VNF lifecycle operation.

5.4.17.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}/cancel

This resource shall support the resource URI variables defined in table 5.4.17.2-1.

Table 5.4.17.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfLcmOpOccId	Identifier of a VNF lifecycle management operation occurrence to be cancelled. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a PATCH or POST request triggering a VNF LCM operation. It can also be retrieved from the "vnfLcmOpOccId" attribute in the VnfLcmOperationOccurrenceNotification.	

5.4.17.3   Resource methods

5.4.17.3.1   POST

The POST method initiates cancelling an ongoing VNF lifecycle operation while it is being executed or rolled back, i.e. the related "VNF LCM operation occurrence" resource is either in "PROCESSING" or "ROLLING_BACK" state.

This method shall follow the provisions specified in the tables 5.4.17.3.1-1 and 5.4.17.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.17.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.17.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	CancelMode	1		The POST request to this resource shall include a CancelMode structure in the payload body to choose between "graceful" and "forceful" cancellation.

	Data type	Cardinality	Response Codes	Description
Response body	n/a		202 Accepted	The request was accepted for processing, but processing has not been completed. The response shall have an empty payload body.
	ProblemDetails	0..1	404 Not Found	Error: The API producer did not find a current representation for the target resource or is not willing to disclose that one exists. The general cause for this error and its handling is specified in clause 4.3.5.4, including rules for the presence of the response body. Specifically in case of this task resource, the response code 404 shall also be returned if the task is not supported for the VNF LCM operation occurrence represented by the parent resource, which means that the task resource consequently does not exist. In this case, the response body shall be present, and shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the VNF LCM operation occurrence resource. Typically, this is due to the fact that the operation occurrence is not in STARTING, PROCESSING or ROLLING_BACK state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.17.3.2   GET

Not supported.

5.4.17.3.3   PUT

Not supported.

5.4.17.3.4   PATCH

Not supported.

5.4.17.3.5   DELETE

Not supported.

5.4.18   Resource: Subscriptions

5.4.18.1   Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF lifecycle management, and to query its subscriptions.

5.4.18.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/subscriptions

This resource shall support the resource URI variables defined in table 5.4.18.2-1.

Table 5.4.18.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

5.4.18.3   Resource methods

5.4.18.3.1   POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 5.4.18.3.1-1 and 5.4.18.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the EM or VNF, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 5.4.18.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.18.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	LccnSubscriptionRequest	1	Details of the subscription to be created, as defined in clause 5.5.2.15.	

Response body	Data type	Cardinality	Response Codes	Description
	LccnSubscription	1	201 Created	The subscription was created successfully. The response body shall contain a representation of the created subscription resource. The HTTP response shall include a "Location:" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.18.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 5.4.18.3.2-1 and 5.4.18.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.18.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. All attribute names that appear in the LccnSubscription and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 5.4.18.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	LccnSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain the representations of all active subscriptions of the functional block that invokes the method.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.18.3.3 PUT

Not supported.

5.4.18.3.4 PATCH

Not supported.

5.4.18.3.5 DELETE

Not supported.

5.4.19 Resource: Individual subscription

5.4.19.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to VNF lifecycle management.

5.4.19.2 Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 5.4.19.2-1.

Table 5.4.19.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource, It can also be retrieved from the "id" attribute in the payload body of that response.	

5.4.19.3   Resource methods

5.4.19.3.1   POST

Not supported.

5.4.19.3.2   GET

The GET method retrieves information about a subscription by reading an individual subscription resource.

This method shall follow the provisions specified in the tables 5.4.19.3.2-1 and 5.4.19.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.19.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.19.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	LccnSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.19.3.3   PUT

Not supported.

5.4.19.3.4   PATCH

Not supported.

5.4.19.3.5   DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 5.4.19.3.5-1 and 5.4.19.3.3-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.19.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.19.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**	
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

5.4.20   Resource: Notification endpoint

5.4.20.1   Description

This resource represents a notification endpoint. The API producer can use this resource to send notifications related to VNF lifecycle changes to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

5.4.20.2   Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 5.4.20.2-1.

Table 5.4.20.2-1: Resource URI variables for this resource

Name	Definition
none supported	

5.4.20.3   Resource methods

5.4.20.3.1   POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 5.4.20.3.1-1 and 5.4.20.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.20.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one of the alternatives defined in table 5.4.20.3.1-2.

Table 5.4.20.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description
Request body	VnfLcmOperationOccurrenceNotification	1	A notification about lifecycle changes triggered by a VNF LCM operation occurrence.
	VnfIdentifierCreationNotification	1	A notification about the creation of a VNF identifier and the related VNF instance resource.
	VnfIdentifierDeletionNotification	1	A notification about the deletion of a VNF identifier and the related VNF instance resource.

	Data type	Cardinality	Response Codes	Description
Response body	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned. The VNFM shall not retry sending the notification if a 4xx/5xx error code other than 401 is received.

5.4.20.3.2   GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 5.4.20.3.2-1 and 5.4.20.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.20.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.20.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			

	Data type	Cardinality	Response Codes	Description
Response body	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

5.4.20.3.3   PUT

Not supported.

5.4.20.3.4   PATCH

Not supported.

5.4.20.3.5   DELETE

Not supported.

5.5   Data model

5.5.1   Introduction

This clause defines the request and response data structures of the VNF Lifecycle management interface. If a request or response contains attributes not defined in the present specification, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

5.5.2   Resource and notification data types

5.5.2.1   Introduction

This clause defines the data structures to be used in resource representations and notifications.

5.5.2.2   Type: VnfInstance

This type represents a VNF instance. It shall comply with the provisions defined in table 5.5.2.2-1.

NOTE: Clause B.3.2 in Annex B provides examples illustrating the relationship among the different run-time information elements (CP, VL and link ports) used to represent the connectivity of a VNF.

Table 5.5.2.2-1: Definition of the VnfInstance data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the VNF instance.
vnfInstanceName	String	0..1	Name of the VNF instance. This attribute can be modified with the PATCH method.
vnfInstanceDescription	String	0..1	Human-readable description of the VNF instance. This attribute can be modified with the PATCH method.
vnfdId	Identifier	1	Identifier of the VNFD on which the VNF instance is based.
vnfProvider	String	1	Provider of the VNF and the VNFD. The value is copied from the VNFD.
vnfProductName	String	1	Name to identify the VNF Product. The value is copied from the VNFD.
vnfSoftwareVersion	Version	1	Software version of the VNF. The value is copied from the VNFD.
vnfdVersion	Version	1	Identifies the version of the VNFD. The value is copied from the VNFD.
vnfPkgId	Identifier	1	Identifier of information held by the NFVO about the specific VNF package on which the VNF is based. This identifier was allocated by the NFVO. This attribute can be modified with the PATCH method. See note 1.
vnfConfigurableProperties	KeyValuePairs	0..1	Current values of the configurable properties of the VNF instance. Configurable properties referred in this attribute are declared in the VNFD (see notes 2 and note 3). These configurable properties include the following standard attributes, which are declared in the VNFD if auto-scaling and/or auto-healing are supported by the VNF: isAutoscaleEnabled: If present, the VNF supports auto-scaling. If set to true, auto-scaling is currently enabled. If set to false, auto-scaling is currently disabled.

Attribute name	Data type	Cardinality	Description
			isAutohealEnabled: If present, the VNF supports auto-healing. If set to true, auto-healing is currently enabled. If set to false, auto-healing is currently disabled. This attribute can be modified with the PATCH method.
instantiationState	Enum (inlined)	1	The instantiation state of the VNF. Permitted values: - NOT_INSTANTIATED: The VNF instance is terminated or not instantiated. - INSTANTIATED: The VNF instance is instantiated.
instantiatedVnfInfo	Structure (inlined)	0..1	Information specific to an instantiated VNF instance. This attribute shall be present if the instantiateState attribute value is INSTANTIATED.
>flavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour applied to this VNF instance.
>vnfState	VnfOperationalStateType	1	State of the VNF instance.
>scaleStatus	ScaleInfo	0..N	Scale status of the VNF, one entry per aspect. Represents for every scaling aspect how "big" the VNF has been scaled w.r.t. that aspect. This attribute shall be present if the VNF supports scaling. See clause B.2 for an explanation of VNF scaling.
>extCpInfo	Structure (inlined)	1..N	Information about the external CPs exposed by the VNF instance.
>id	IdentifierInVnf	1	Identifier of the external CP instance and the related information instance.
>>cpdId	IdentifierInVnfd	1	Identifier of the external CPD, VnfExtCpd, in the VNFD.
>>cpProtocolInfo	CpProtocolInfo	0..N	Network protocol information for this CP.
>>extLinkPortId	Identifier	0..1	Identifier of the "extLinkPortInfo" structure inside the the "extVirtualLinkInfo" structure. Shall be present if the CP is associated to a link port.
>extVirtualLinkInfo	ExtVirtualLinkInfo	0..N	Information about the external VLs the VNF instance is connected to.
>extManagedVirtualLinkInfo	ExtManagedVirtualLinkInfo	0..N	Information about the externally managed internal VLs of the VNF instance.
>monitoringParameters	MonitoringParameter	0..N	Active monitoring parameters.
>localizationLanguage	String	0..1	Information about localization language of the VNF (includes e.g. strings in the VNFD). The localization languages supported by a VNF can be declared in the VNFD, and localization language selection can take place at instantiation time. The value shall comply with the format defined in IETF RFC 5646 [7].
>vnfcResourceInfo	VnfcResourceInfo	0..N	Information about the virtualised compute and storage resources used by the VNFCs of the VNF instance.
>vnfVirtualLinkResourceInfo	VnfVirtualLinkResourceInfo	0..N	Information about the virtualised network resources used by the VLs of the VNF instance.
>virtualStorageResourceInfo	VirtualStorageResourceInfo	0..N	Information about the virtualised storage resources used as storage for the VNF instance.
>vnfcInfo	VnfcInfo	0..N	Information about the VNFC instances.
metadata	KeyValuePairs	0..1	Additional VNF-specific metadata describing the VNF instance.Metadata that are writeable are declared in the VNFD (see note 2). This attribute can be modified with the PATCH method.
extensions	KeyValuePairs	0..1	VNF specific attributes that affect the lifecycle management of this VNF instance by the VNFM, or the lifecycle management scripts. Extensions that are writeable are declared in the VNFD (see note 2). This attribute can be modified with the PATCH method.
_links	Structure (inlined)	1	Links to resources related to this resource

Attribute name	Data type	Cardinality	Description
>self	Link	1	URI of this resource
>indicators	Link	0..1	Indicators related to this VNF instance, if applicable
>instantiate	Link	0..1	Link to the "instantiate" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance in NOT_INSTANTIATED state).
>terminate	Link	0..1	Link to the "terminate" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>scale	Link	0..1	Link to the "scale" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>scaleToLevel	Link	0..1	Link to the "scale_to_level" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>changeFlavour	Link	0..1	Link to the "change_flavour" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>heal	Link	0..1	Link to the "heal" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>operate	Link	0..1	Link to the "operate" task resource, if the related operation is supported for this VNF instance, and is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
>changeExtConn	Link	0..1	Link to the "change_ext_conn" task resource, if the related operation is possible based on the current status of this VNF instance resource (i.e. VNF instance is in INSTANTIATED state).
NOTE 1: Modifying the value of this attribute shall not be performed when conflicts exist between the previous and the newly referred VNF package, i.e. when the new VNFD is not changed with respect to the previous VNFD apart from referencing to other VNF software images. In order to avoid misalignment of the VnfInstance with the current VNF's on-boarded VNF package, the values of attributes in the VnfInstance that have corresponding attributes in the VNFD shall be kept in sync with the values in the VNFD.			
NOTE 2: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			
NOTE 3: VNF configurable properties are sometimes also referred to as configuration parameters applicable to a VNF. Some of these are set prior to instantiation and cannot be modified if the VNF is instantiated, some are set prior to instantiation (are part of initial configuration) and can be modified later, and others can be set only after instantiation. The applicability of certain configuration may depend on the VNF and the required operation of the VNF at a certain point in time.			

5.5.2.3   Type: CreateVnfRequest

This type represents a request parameters for the "Create VNF identifier" operation. It shall comply with the provisions defined in table 5.5.2.3-1.

Table 5.5.2.3-1: Definition of the CreateVnfRequest data type

Attribute name	Data type	Cardinality	Description
vnfdId	Identifier	1	Identifier that identifies the VNFD which defines the VNF instance to be created.
vnfInstanceName	String	0..1	Human-readable name of the VNF instance to be created.
vnfInstanceDescription	String	0..1	Human-readable description of the VNF instance to be created.

5.5.2.4   Type: InstantiateVnfRequest

This type represents request parameters for the "Instantiate VNF" operation It shall comply with the provisions defined in table 5.5.2.4-1.

Table 5.5.2.4-1: Definition of the InstantiateVnfRequest data type

Attribute name	Data type	Cardinality	Description
flavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour to be instantiated.
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated.
extVirtualLinks	ExtVirtualLinkData	0..N	Information about external VLs to connect the VNF to.
extManagedVirtualLinks	ExtManagedVirtualLinkData	0..N	Information about internal VLs that are managed by other entities than the VNFM. See note.
localizationLanguage	String	0..1	Localization language of the VNF to be instantiated. The value shall comply with the format defined in IETF RFC 5646 [7].
additionalParams	KeyValuePairs	0..1	Additional input parameters for the instantiation process, specific to the VNF being instantiated as declared in the VNFD as part of "InstantiateVnfOpConfig".
NOTE: The present document assumes that externally-managed internal VLs are managed by the NFVO and created towards the VIM.			

5.5.2.5   Type: ScaleVnfRequest

This type represents request parameters for the "Scale VNF" operation. It shall comply with the provisions defined in table 5.5.2.5-1. See clause B.2 for an explanation of VNF scaling.

Table 5.5.2.5-1: Definition of the ScaleVnfRequest data type

Attribute name	Data type	Cardinality	Description
type	Enum (inlined)	1	Indicates the type of the scale operation requested. Permitted values: SCALE_OUT: adding additional VNFC instances to the VNF to increase capacity SCALE_IN: removing VNFC instances from the VNF in order to release unused capacity.
aspectId	IdentifierInVnfd	1	Identifier of the scaling aspect.
numberOfSteps	Integer	0..1	Number of scaling steps to be executed as part of this Scale VNF operation. It shall be a positive number and the default value shall be 1.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the scaling process, specific to the VNF being scaled, as declared in the VNFD as part of "ScaleVnfOpConfig".

5.5.2.6   Type: ScaleVnfToLevelRequest

This type represents request parameters for the "Scale VNF to Level" operation. It shall comply with the provisions defined in table 5.5.2.6-1. See clause B.2 in Annex B for an explanation of VNF scaling.

Table 5.5.2.6-1: Definition of the ScaleVnfToLevelRequest data type

Attribute name	Data type	Cardinality	Description
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the target instantiation level of the current deployment flavour to which the VNF is requested to be scaled. See note.
scaleInfo	ScaleInfo	0..N	For each scaling aspect of the current deployment flavour, indicates the target scale level to which the VNF is to be scaled. See note.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the scaling process, specific to the VNF being scaled, as declared in the VNFD as part of "ScaleVnfToLevelOpConfig".
NOTE: Either the instantiationLevelId attribute or the scaleInfo attribute shall be included.			

5.5.2.7   Type: ChangeVnfFlavourRequest

This type represents request parameters for the "Change VNF flavour" operation. It shall comply with the provisions defined in table 5.5.2.7-1.

Table 5.5.2.7-1: Definition of the ChangeVnfFlavourRequest data type

Attribute name	Data type	Cardinality	Description
newFlavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour to be instantiated.
instantiationLevelId	IdentifierInVnfd	0..1	Identifier of the instantiation level of the deployment flavour to be instantiated. If not present, the default instantiation level as declared in the VNFD is instantiated.
extVirtualLinks	ExtVirtualLinkData	0..N	Information about external VLs to connect the VNF to.
extManagedVirtualLinks	ExtManagedVirtualLinkData	0..N	Information about internal VLs that are managed by other entities than the VNFM. See note.
additionalParams	KeyValuePairs	0..1	Additional input parameters for the flavour change process, specific to the VNF being modified, as declared in the VNFD as part of "ChangeVnfFlavourOpConfig".
NOTE: The present document assumes that externally-managed internal VLs are managed by the NFVO and created towards the VIM.			

5.5.2.8   Type: TerminateVnfRequest

This type represents request parameters for the "Terminate VNF" operation. It shall comply with the provisions defined in table 5.5.2.8-1.

Table 5.5.2.8-1: Definition of the TerminateVnfRequest data type

Attribute name	Data type	Cardinality	Description
terminationType	Enum (inlined)	1	Indicates whether forceful or graceful termination is requested. See note. Permitted values: FORCEFUL: The VNFM will shut down the VNF and release the resources immediately after accepting the request.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the termination process, specific to the VNF being terminated, as declared in the VNFD as part of "TerminateVnfOpConfig".
NOTE:	In case of forceful termination, the VNF instance is terminated immediately. If the VNF is still in service, this can adversely impact the network service, and therefore, the EM needs to determine if forceful termination is applicable in the particular situation.		

5.5.2.9   Type: HealVnfRequest

This type represents request parameters for the "Heal VNF" operation. It shall comply with the provisions defined in table 5.5.2.9-1.

Table 5.5.2.9-1: Definition of the HealVnfRequest data type

Attribute name	Data type	Cardinality	Description
vnfcInstanceId	Identifier	0..N	List of VNFC instances requiring a healing action.
cause	String	0..1	Indicates the reason why a healing procedure is required.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the healing process, specific to the VNF being healed as declared in the VNFD as part of "HealVnfOpConfig".
healScript	String	0..1	Provides link to a script that should be executed as part of the healing action or a set of rules for healing procedure.

5.5.2.10   Type: OperateVnfRequest

This type represents request parameters for the "Operate VNF" operation. It shall comply with the provisions defined in table 5.5.2.10-1.

Table 5.5.2.10-1: Definition of the OperateVnfRequest data type

Attribute name	Data type	Cardinality	Description
vnfcInstanceId	Identifier	0..N	Identifier of VNFC instances. Cardinality can be "0" to denote that the request applies to the whole VNF and not a specific VNFC instance.
changeStateTo	VnfOperationalState Type	1	The desired operational state (i.e. started or stopped) to change the VNF/VNFC to.
stopType	Enum (inlined)	0..1	It signals whether forceful or graceful stop is requested. Ignored if changeStateTo=STARTED. Permitted values: FORCEFUL: The VNFM will stop down the VNF or the affected VNFCs immediately after accepting the request.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the process, specific to the VNF of which the operation status is changed, as declared in the VNFD as part of "OperateVnfOpConfig".

5.5.2.11   Type: ChangeExtVnfConnectivityRequest

This type represents request parameters for the "Change external VNF connectivity" operation to modify the external connectivity of a VNF instance. It shall comply with the provisions defined in table 5.5.2.11-1.

Table 5.5.2.11-1: Definition of the ChangeExtVnfConnectivityRequest data type

Attribute name	Data type	Cardinality	Description
extVirtualLinks	ExtVirtualLinkData	1..N	Information about external VLs to change (e.g. connect the VNF to).
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the EM as input to the process, specific to the VNF of which the external connectivity is changed, as declared in the VNFD as part of "ChangeExtVnfConnectivityOpConfig".

The following behaviour applies for the changes that can be performed with this operation:

- To change the connection of external CP instances based on certain external CPDs from a "source" external VL to a different "target" external VL, the identifier of the "target" external VL shall be sent in the "extVirtualLinkId" attribute of the "extVirtualLinks" parameter, and the "extCps" attributes of that parameter shall refer via the "cpdId" attribute to the external CPDs of the corresponding external connection point instances that are to be reconnected to the target external VL.

    NOTE: This means that all CP instances based on a given external CPD will be reconnected. See clause B.3.3 in Annex B for an illustration.

- To change the connectivity parameters of the external CPs connected to a particular external VL, including changing addresses, the identifier of that external VL shall be sent in the "extVirtualLinkId" attribute of the "extVirtualLinks" parameter, and the "extCps" attribute of that parameter shall contain at least those entries with modified parameters.

5.5.2.12   Type: VnfInfoModificationRequest

This type represents attribute modifications for an "Individual VNF instance" resource, i.e. modifications to a resource representation based on the "VnfInstance" data type. The attributes of "VnfInstance" that can be modified according to the provisions in clause 5.5.2.2 are included in the "VnfInfoModificationRequest" data type.

The "VnfInfoModificationRequest" data type shall comply with the provisions defined in table 5.5.2.12-1.

Table 5.5.2.12-1: Definition of the VnfInfoModificationRequest data type

Attribute name	Data type	Cardinality	Description
vnfInstanceName	String	0..1	New value of the "vnfInstanceName" attribute in "VnfInstance", or "null" to remove the attribute.
vnfInstanceDescription	String	0..1	New value of the "vnfInstanceDescription" attribute in "VnfInstance", or "null" to remove the attribute.
vnfPkgId	Identifier	0..1	New value of the "vnfPkgId" attribute in "VnfInstance". The value "null" is not permitted.
vnfConfigurableProperties	KeyValuePairs	0..1	Modifications of the "vnfConfigurableProperties" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).
metadata	KeyValuePairs	0..1	Modifications of the "metadatta" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).
extensions	KeyValuePairs	0..1	Modifications of the "extensions" attribute in "VnfInstance". If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).
vnfcInfoModifications	VnfcInfoModifications	0..N	Modifications of certain entries in the "vnfcInfo" attribute array in the "instantiatedVnfInfo" attribute of "VnfInstance".

The following provisions shall apply when modifying an attribute that is an array of objects of type "VnfcInfo" by supplying an array of objects of type "VnfcInfoModifications".

Assumptions:

1) "oldList" is the "VnfcInfo" array to be modified and "newList" is the "VnfcInfoModifications" array that contains the changes.

2) "oldEntry" is an entry in "oldList" and "newEntry" is an entry in "newList".

3) A "newEntry" has a "corresponding entry" if there exists an "oldEntry" that has the same content of the "id" attribute as the "newEntry"; a "newEntry" has no corresponding entry if no such "oldEntry" exists.

4) In any array of "VnfcInfo" resp. "VnfcInfoModifications" structures, the content of "id" is unique (i.e. there shall be no two entries with the same content of "id").

Provisions:

1) If any "newEntry" in "newList" has no corresponding entry in "oldList", the "VnfInfoModifications" data structure shall be rejected as erroneous, providing information about the invalid id values.

 NOTE: New entries in the "vnfcInfo" array can only be created by the VNFM.

2) For each "newEntry" in "newList" that has a corresponding "oldEntry" in "oldList", the value of "oldEntry" shall be updated with the value of "newEntry" according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).

5.5.2.12a   Type: VnfInfoModifications

This type represents attribute modifications that were performed on an "Individual VNF instance" resource. The attributes that can be included consist of those requested to be modified explicitly in the "VnfInfoModificationRequest" data structure, and additional attributes of the "VnfInstance" data structure that were modified implicitly e.g. when modifying the referenced VNF package.

The "VnfInfoModifications" data type shall comply with the provisions defined in table 5.5.2.12a-1.

Table 5.5.2.12a-1: Definition of the VnfInfoModifications data type

Attribute name	Data type	Cardinality	Description
vnfInstanceName	String	0..1	If present, this attribute signals modifications of the "vnfInstanceName" attribute in "VnfInstance" as defined in clause 5.5.2.12.
vnfInstanceDescription	String	0..1	If present, this attribute signals modifications of the "vnfInstanceDescription" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vnfConfigurableProperties	KeyValuePairs	0..1	If present, this attribute signals modifications of the "vnfConfigurableProperties" attribute in "VnfInstance", as defined in clause 5.5.2.12.
metadata	KeyValuePairs	0..1	If present, this attribute signals modifications of the "metadata" attribute in "VnfInstance", as defined in clause 5.5.2.12.
extensions	KeyValuePairs	0..1	If present, this attribute signals modifications of the "extensions" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vnfPkgId	Identifier	0..1	If present, this attribute signals modifications of the "vnfPkgId" attribute in "VnfInstance", as defined in clause 5.5.2.12.
vnfdId	Identifier	0..1	If present, this attribute signals modifications of the "vnfdId" attribute in "VnfInstance". See note.
vnfProvider	String	0..1	If present, this attribute signals modifications of the "vnfProvider" attribute in "VnfInstance". See note.
vnfProductName	String	0..1	If present, this attribute signals modifications of the "vnfProductName" attribute in "VnfInstance". See note.
vnfSoftwareVersion	Version	0..1	If present, this attribute signals modifications of the "vnfSoftwareVersion" attribute in "VnfInstance". See note.
vnfdVersion	Version	0..1	If present, this attribute signals modifications of the "vnfdVersion" attribute in "VnfInstance". See note.
NOTE: If present, this attribute (which depends on the value of the "vnfPkgId" attribute) was modified implicitly following a request to modify the "vnfPkgId" attribute, by copying the value of this attribute from the VNFD in the VNF Package identified by the "vnfPkgId" attribute.			

5.5.2.13   Type: VnfLcmOpOcc

This type represents a VNF lifecycle management operation occurrence. It shall comply with the provisions defined in table 5.5.2.13-1.

Table 5.5.2.13-1: Definition of the VnfLcmOpOcc data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this VNF lifecycle management operation occurrence.
operationState	LcmOperationStateType	1	The state of the LCM operation.
stateEnteredTime	DateTime	1	Date-time when the current state was entered.
startTime	DateTime	1	Date-time of the start of the operation.
vnfInstanceId	Identifier	1	Identifier of the VNF instance to which the operation applies.
grantId	Identifier	0..1	Identifier of the grant related to this VNF LCM operation occurrence, if such grant exists.
operation	LcmOperationType	1	Type of the actual LCM operation represented by this VNF LCM operation occurrence.
isAutomaticInvocation	Boolean	1	Set to true if this VNF LCM operation occurrence has been triggered by an automated procedure inside the VNFM (i.e. ScaleVnf/ScaleVnfToLevel triggered by auto-scale, or HealVnf triggered by auto-heal). Set to false otherwise.

Attribute name	Data type	Cardinality	Description
operationParams	Object	1	Input parameters of the LCM operation. This attribute shall be formatted according to the request data type of the related LCM operation. The following mapping between operationType and the data type of this attribute shall apply: - INSTANTIATE: InstantiateVnfRequest - SCALE: ScaleVnfRequest - SCALE_TO_LEVEL: ScaleVnfToLevelRequest - CHANGE_FLAVOUR: ChangeVnfFlavourRequest - OPERATE: OperateVnfRequest - HEAL: HealVnfRequest - CHANGE_EXT_CONN: ChangeExtVnfConnectivityRequest - TERMINATE: TerminateVnfRequest - MODIFY_INFO: VnfInfoModificationRequest
isCancelPending	Boolean	1	If the VNF LCM operation occurrence is in "STARTING", "PROCESSING" or "ROLLING_BACK" state and the operation is being cancelled, this attribute shall be set to true. Otherwise, it shall be set to false.
cancelMode	CancelModeType	0..1	The mode of an ongoing cancellation. Shall be present when isCancelPending=true, and shall be absent otherwise.
error	ProblemDetails	0..1	If "operationState" is "FAILED_TEMP" or "FAILED" or "operatioState" is "PROCESSING" or "ROLLING_BACK" and previous value of "operationState" was "FAILED_TEMP", this attribute shall be present and contain error information unless it has been requested to be excluded via an attribute selector.
resourceChanges	Structure (inlined)	0..1	This attribute contains information about the cumulative changes to virtualised resources that were performed so far by the LCM operation since its start, if applicable.
>affectedVnfcs	AffectedVnfc	0..N	Information about VNFC instances that were affected during the lifecycle operation. See note.
>affectedVirtualLinks	AffectedVirtualLink	0..N	Information about VL instances that were affected during the lifecycle operation. See note.
>affectedVirtualStorages	AffectedVirtualStorage	0..N	Information about virtualised storage instances that were affected during the lifecycle operation. See note.
changedInfo	VnfInfoModifications	0..1	Information about the changed VNF instance information, including VNF configurable properties, if applicable. See note.
changedExtConnectivity	ExtVirtualLinkInfo	0..N	Information about changed external connectivity, if applicable. See note.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>vnfInstance	Link	1	Link to the VNF instance that the operation applies to.
>grant	Link	0..1	Link to the grant for this operation, if one exists.
>cancel	Link	0..1	Link to the task resource that represents the "cancel" operation for this VNF LCM operation occurrence, if cancelling is currently allowed.
>retry	Link	0..1	Link to the task resource that represents the "retry" operation for this VNF LCM operation occurrence, if retrying is currently allowed.
>rollback	Link	0..1	Link to the task resource that represents the "rollback" operation for this VNF LCM operation occurrence, if rolling back is currently allowed.
>fail	Link	0..1	Link to the task resource that represents the "fail" operation for this VNF LCM operation occurrence, if declaring as failed is currently allowed.
NOTE: This allows the consumer to obtain the information contained in the latest "result" notification if it has not received it due to an error or a wrongly configured subscription filter.			

5.5.2.14   Type: CancelMode

This type represents a parameter to select the mode of cancelling an ongoing VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.2.14-1.

Table 5.5.2.14-1: Definition of the CancelMode data type

Attribute name	Data type	Cardinality	Description
cancelMode	CancelModeType	1	Cancellation mode to apply.

5.5.2.15   Type: LccnSubscriptionRequest

This type represents a subscription request related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.2.15-1.

Table 5.5.2.15-1: Definition of the LccnSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	LifecycleChangeNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

5.5.2.16   Type: LccnSubscription

This type represents a subscription related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.2.16-1.

Table 5.5.2.16-1: Definition of the LccnSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	LifecycleChangeNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.

5.5.2.17   Type: VnfLcmOperationOccurrenceNotification

This type represents a VNF lifecycle management operation occurrence notification, which informs the receiver of changes in the VNF lifecycle caused by a VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.2.17-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM produced when there is a change in the VNF lifecycle caused by VNF LCM operation occurrence including:

- Instantiation of the VNF

- Scaling of the VNF instance (including auto-scaling)

- Healing of the VNF instance (including auto-healing)

- Change of the state of the VNF instance (i.e. Operate VNF)

- Change of the deployment flavour of the VNF instance

- Change of the external connectivity of the VNF instance

- Termination of the VNF instance

- Modification of VNF instance information and/or VNF configurable properties through the "PATCH" method on the "Individual VNF instance" resource

If this is the initial notification about the start of a VNF LCM operation occurrence, it is assumed that the notification is sent by the VNFM before any action (including sending the grant request) is takenas part of the LCM operation. Due to possible race conditions, the "start" notification, the grant request and the LCM operation acknowledgment can arrive in any order at the consumer, and the consumer shall be able to handle such a situation.

If this is a notification about a final or intermediate result state of a VNF LCM operation occurrence, the notification shall be sent after all related actions of the LCM operation that led to this state have been executed.

The new state shall be set in the VnfLcmOpOcc resource before the notification about the state change is sent.

See clause 5.6.2.2 for further provisions regarding sending this notification in case of handling LCM operation errors.

Table 5.5.2.17-1: Definition of the VnfLcmOperationOccurrenceNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfLcmOperationOccurrenceNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
notificationStatus	Enum (inlined)	1	Indicates whether this notification reports about the start of a lifecycle operation or the result of a lifecycle operation. Permitted values: - START: Informs about the start of the VNF LCM operation occurrence. - RESULT: Informs about the final or intermediate result of the VNF LCM operation occurrence.
operationState	LcmOperationStateType	1	The state of the VNF LCM operation occurrence.
vnfInstanceId	Identifier	1	The identifier of the VNF instance affected.
operation	LcmOperationType	1	The lifecycle management operation.
isAutomaticInvocation	Boolean	1	Set to true if this VNF LCM operation occurrence has been triggered by an automated procedure inside the VNFM (i.e. ScaleVnf/ScaleVnfToLevel triggered by auto-scale, or HealVnf triggered by auto-heal). Set to false otherwise.
vnfLcmOpOccId	Identifier	1	The identifier of the VNF lifecycle management operation occurrence associated to the notification.
affectedVnfcs	AffectedVnfc	0..N	Information about VNFC instances that were affected during the lifecycle operation. See note.
affectedVirtualLinks	AffectedVirtualLink	0..N	Information about VL instances that were affected during the lifecycle operation. See note.
affectedVirtualStorages	AffectedVirtualStorage	0..N	Information about virtualised storage instances that were affected during the lifecycle operation. See note.
changedInfo	VnfInfoModifications	0..1	Information about the changed VNF instance information, including changed VNF configurable properties. Shall be present if the "notificationStatus" is set to "RESULT" and the operation has performed any

Attribute name	Data type	Cardinality	Description
			changes to VNF instance information, including VNF configurable properties. Shall be absent otherwise.
changedExtConnectivity	ExtVirtualLinkInfo	0..N	Information about changed external connectivity, if this notification represents the result of a lifecycle operation occurrence. Shall be present if the "notificationStatus" is set to "RESULT" and the "operation" is set to "CHANGE_EXT_CONN". Shall be absent otherwise.
error	ProblemDetails	0..1	Details of the latest error, if one has occurred during executing the VNF LCM operation (see clause 4.3.5). Shall be present if the "operationState" attribute is "FAILED_TEMP" or "FAILED", and shall be absent otherwise.
_links	LccnLinks	1	Links to resources related to this notification.
NOTE:	Shall be present if the "notificationStatus" is set to "RESULT" and the operation has performed any resource modification. Shall be absent otherwise. This attribute contains information about the cumulative changes to virtualised resources that were performed so far by the VNF LCM operation occurrence and by any of the error handling procedures for that operation occurrence.		

5.5.2.18   Type: VnfIdentifierCreationNotification

This type represents a VNF identifier creation notification, which informs the receiver of the creation of a new VNF instance resource and the associated VNF instance identifier. It shall comply with the provisions defined in table 5.5.2.18-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM when it has created a VNF instance resource and the associated VNF instance identifier.

Table 5.5.2.18-1: Definition of the VnfIdentifierCreationNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIdentifierCreationNotification" for this notification type.
subscriptionId	Identifier	0..1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfInstanceId	Identifier	1	The created VNF instance identifier.
_links	LccnLinks	1	Links to resources related to this notification.

5.5.2.19   Type: VnfIdentifierDeletionNotification

This type represents a VNF identifier deletion notification, which informs the receiver of the deletion of a new VNF instance resource and the associated VNF instance identifier. It shall comply with the provisions defined in table 5.5.2.19-1. The support of the notification is mandatory.

This notification shall be triggered by the VNFM when it has deleted a VNF instance resource and the associated VNF instance identifier.

Table 5.5.2.19-1: Definition of the VnfIdentifierDeletionNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIdentifierDeletionNotification" for this notification type.
subscriptionId	Identifier	0..1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfInstanceId	Identifier	1	The deleted VNF instance identifier.
_links	LccnLinks	1	Links to resources related to this notification.

5.5.3　　Referenced structured data types

5.5.3.1　　Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

5.5.3.2　　Type: ExtVirtualLinkData

This type represents an external VL. It shall comply with the provisions defined in table 5.5.3.2-1.

Table 5.5.3.2-1: Definition of the ExtVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	The identifier of the external VL instance, if provided. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable. See note.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of this resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	The identifier of the resource in the scope of the VIM or the resource provider.
extCps	VnfExtCpData	1..N	External CPs of the VNF to be connected to this external VL.
extLinkPorts	ExtLinkPortData	0..N	Externally provided link ports to be used to connect external connection points to this external VL. If this attribute is not present, the VNFM shall create the link ports on the external VL.
NOTE: The information about the VIM connection referenced by the VIM connection id is known to the VNFM. Moreover, the identifier of the VIM connection provides scope to the resourceId.			

5.5.3.3　　Type: ExtVirtualLinkInfo

This type represents information about an external VL. It shall comply with the provisions defined in table 5.5.3.3-1.

ETSI

Table 5.5.3.3-1: Definition of the ExtVirtualLinkInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the external VL and the related external VL information instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
resourceHandle	ResourceHandle	1	Reference to the resource realizing this VL.
linkPorts	ExtLinkPortInfo	0..N	Link ports of this VL.

5.5.3.4   Type: ExtManagedVirtualLinkData

This type represents an externally-managed internal VL. It shall comply with the provisions defined in table 5.5.3.4-1.

Table 5.5.3.4-1: Definition of the ExtManagedVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	The identifier of the externally-managed internal VL instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
virtualLinkDescId	IdentifierInVnfd	1	The identifier of the VLD in the VNFD for this VL.
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable. See note.
resourceProviderId	Identifier	0..1	Identifies the entity responsible for the management of this resource. This attribute shall only be supported and present if VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	The identifier of the resource in the scope of the VIM or the resource provider.
NOTE:	The information about the VIM connection referenced by the VIM connection id is known to the VNFM. Moreover, the identifier of the VIM connection provides scope to the resourceId.		

5.5.3.5   Type: ExtManagedVirtualLinkInfo

This type provides information about an externally-managed virtual link. It shall comply with the provisions defined in table 5.5.3.5-1.

Table 5.5.3.5-1: Definition of the ExtManagedVirtualLinkInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the externally-managed internal VL and the related externally-managed VL information instance. The identifier is assigned by the NFV-MANO entity that manages this VL instance.
vnfVirtualLinkDescId	IdentifierInVnfd	1	Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource.
vnfLinkPorts	VnfLinkPortInfo	0..N	Link ports of this VL.

5.5.3.6   Type: VnfExtCpData

This type represents configuration information for external CPs created from a CPD. It shall comply with the provisions defined in table 5.5.3.6-1.

Table 5.5.3.6-1: Definition of the VnfExtCpData data type

Attribute name	Data type	Cardinality	Description
cpdId	IdentifierInVnfd	1	The identifier of the CPD in the VNFD.
cpConfig	VnfExtCpConfig	1..N	List of instance data that need to be configured on the CP instances created from the respective CPD.

5.5.3.6a   Type: VnfExtCpConfig

This type represents an externally provided link port or network address information per instance of an external connection point. In case a link port is provided, the VNFM shall use that link port when connecting the external CP to the external VL. In a link port is not provided, the VNFM shall create a link port on the external VL, and use that link port to connect the external CP to the external VL.

This type shall comply with the provisions defined in table 5.5.3.6a-1.

Table 5.5.3.6a-1: Definition of the VnfExtCpConfig data type

Attribute name	Data type	Cardinality	Description
cpInstanceId	IdentifierInVnf	0..1	Identifier of the external CP instance to which this set of configuration parameters is requested to be applied. Shall be present if this instance has already been created.
linkPortId	Identifier	0..1	Identifier of a pre-configured link port to which the external CP will be associated. See note.
cpProtocolData	CpProtocolData	0..N	Parameters for configuring the network protocols on the link port that connects the CP to a VL. See note.
NOTE: The following conditions apply to the attributes "linkPortId" and " cpProtocolData": 1) The "linkPortId" and "cpProtocolData" attributes shall both be absent for the deletion of an existing external CP instance addressed by cpInstanceId. 2) At least one of these attributes shall be present for a to-be-created external CP instance or an existing external CP instance. 3) If the "linkPortId" attribute is absent, the VNFM shall create a link port. 4) If the "cpProtocolData" attribute is absent, the "linkPortId" attribute shall be provided referencing a pre-created link port, and the VNFM can use means outside the scope of the present document to obtain the pre-configured address information for the connection point from the resource representing the link port. 5) If both "cpProtocolData" and "linkportId" are provided, the API consumer shall ensure that the cpProtocolData can be used with the pre-created link port referenced by "linkPortId".			

5.5.3.6b   Type: CpProtocolData

This type represents network protocol data. It shall comply with the provisions defined in table 5.5.3.6b-1.

Table 5.5.3.6b-1: Definition of the CpProtocolData data type

Attribute name	Data type	Cardinality	Description
layerProtocol	Enum (inlined)	1	Identifier of layer(s) and protocol(s). Permitted values: IP_OVER_ETHERNET See note.
ipOverEthernet	IpOverEthernetAddressData	0..1	Network address data for IP over Ethernet to assign to the extCP instance. Shall be present if layerProtocol is equal to "IP_OVER_ETHERNET", and shall be absent otherwise.
NOTE: This attribute allows to signal the addition of further types of layer and protocol in future versions of the present document in a backwards-compatible way. In the current version of the present document, only IP over Ethernet is supported.			

5.5.3.6c   Type: IpOverEthernetAddressData

This type represents network address data for IP over Ethernet. It shall comply with the provisions defined in table 5.5.3.6c-1.

Table 5.5.3.6c-1: Definition of the IpOverEthernetAddressData data type

Attribute name	Data type	Cardinality	Description
macAddress	MacAddress	0..1	MAC address. If this attribute is not present, it shall be chosen by the VIM. See note 1.
ipAddresses	Structure (inlined)	0..N	List of IP addresses to assign to the CP instance. Each entry represents IP address data for fixed or dynamic IP address assignment per subnet. If this attribute is not present, no IP address shall be assigned. See note 1.
>type	Enum (inlined)	1	The type of the IP addresses. Permitted values: IPV4, IPV6.
>fixedAddresses	IpAddress	0..N	Fixed addresses to assign (from the subnet defined by "subnetId" if provided). See note 2.
>numDynamicAddresses	Integer	0..1	Number of dynamic addresses to assign (from the subnet defined by "subnetId" if provided). See note 2.
>addressRange	Structure (inlined)	0..1	An IP address range to be used, e.g. in case of egress connections. In case this attribute is present, IP addresses from the range will be used. See note 2.
>>minAddress	IpAddress	1	Lowest IP address belonging to the range
>>maxAddress	IpAddress	1	Highest IP address belonging to the range
>subnetId	IdentifierInVim	0..1	Subnet defined by the identifier of the subnet resource in the VIM. In case this attribute is present, IP addresses from that subnet will be assigned; otherwise, IP addresses not bound to a subnet will be assigned.
NOTE 1: At least one of "macAddress" or "ipAddresses" shall be present.			
NOTE 2: Exactly one of "fixedAddresses", "numDynamicAddresses" or "ipAddressRange" shall be present.			

5.5.3.7   Type: ScaleInfo

This type represents the scale level of a VNF instance related to a scaling aspect. It shall comply with the provisions defined in table 5.5.3.7-1.

Table 5.5.3.7-1: Definition of the ScaleInfo data type

Attribute name	Data type	Cardinality	Description
aspectId	IdentifierInVnfd	1	Identifier of the scaling aspect.
scaleLevel	Integer	1	Indicates the scale level. The minimum value shall be 0 and the maximum value shall be <= maxScaleLevel as described in the VNFD.

5.5.3.8   Type: VnfcResourceInfo

This type represents the information on virtualised compute and storage resources used by a VNFC in a VNF instance. It shall comply with the provisions defined in table 5.5.3.8-1.

Table 5.5.3.8-1: Definition of the VnfcResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VnfcResourceInfo instance.
vduId	IdentifierInVnfd	1	Reference to the applicable VDU in the VNFD. See note.
computeResource	ResourceHandle	1	Reference to the VirtualCompute resource.
storageResourceIds	IdentifierInVnf	0..N	References to the VirtualStorage resources. The value refers to a VirtualStorageResourceInfo item in the VnfInstance.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
vnfcCpInfo	Structure (inlined)	0..N	CPs of the VNFC instance.
>id	IdentifierInVnf	1	Identifier of this VNFC CP instance and the associated array entry.
>cpdId	IdentifierInVnfd	1	Identifier of the VDU CPD, cpdId, in the VNFD. See note.
>vnfExtCpId	IdentifierInVnf	0..1	When the VNFC CP is exposed as external CP of the VNF, the identifier of this external VNF CP.
>cpProtocolInfo	CpProtocolInfo	0..N	Network protocol information for this CP.
>vnfLinkPortId	IdentifierInVnf	0..1	Identifier of the "vnfLinkPortInfo" structure in the "vnfVirtualLinkResourceInfo" structure. Shall be present if the CP is associated to a link port.
metadata	KeyValuePairs	0..1	Metadata about this resource.
NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			

5.5.3.9   Type: VnfVirtualLinkResourceInfo

This type represents the information that allows addressing a virtualised resource that is used by an internal VL instance in a VNF instance. It shall comply with the provisions defined in table 5.5.3.9-1.

Table 5.5.3.9-1: Definition of the VnfVirtualLinkResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VnfVirtualLinkResourceInfo instance.
vnfVirtualLinkDescId	IdentifierInVnfd	1	Identifier of the VNF Virtual Link Descriptor (VLD) in the VNFD.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
vnfLinkPorts	VnfLinkPortInfo	0..N	Links ports of this VL. Shall be present when the linkPort is used for external connectivity by the VNF (refer to VnfLinkPortInfo). May be present otherwise.
metadata	KeyValuePairs	0..1	Metadata about this resource.

5.5.3.10   Type: VirtualStorageResourceInfo

This type represents the information that allows addressing a virtualised resource that is used by a VNF instance. It shall comply with the provisions defined in table 5.5.3.10-1.

Table 5.5.3.10-1: Definition of the VirtualStorageResourceInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this VirtualStorageResourceInfo instance.
virtualStorageDescId	IdentifierInVnfd	1	Identifier of the VirtualStorageDesc in the VNFD.
storageResource	ResourceHandle	1	Reference to the VirtualStorage resource.
reservationId	Identifier	0..1	The reservation identifier applicable to the resource. It shall be present when an applicable reservation exists.
metadata	KeyValuePairs	0..1	Metadata about this resource.

5.5.3.11   Type: VnfLinkPortInfo

This type represents a link port of an internal VL of a VNF. It shall comply with the provisions defined in table 5.5.3.11-1.

Table 5.5.3.11-1: Definition of the VnfLinkPortInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised network resource realizing this link port.
cpInstanceId	IdentifierInVnf	0..1	When the link port is used for external connectivity by the VNF, this attribute represents the identifier of the external CP of the VNF to be connected to this link port. When the link port is used for internal connectivity in the VNF, this attribute represents the identifier of the VNFC CP to be connected to this link port. There shall be at most one link port associated with any external connection point instance or internal connection point (i.e. VNFC CP) instance. The value refers to an extCpInfo item in the VnfInstance or a "vnfcCpInfo" item of a "vnfcResourceInfo" item in the VnfInstance.

5.5.3.12   Type: ExtLinkPortInfo

This type represents information about a link port of an external VL, i.e. a port providing connectivity for the VNF to an NS VL. It shall comply with the provisions defined in table 5.5.3.12-1.

Table 5.5.3.12-1: Definition of the ExtLinkPortInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised resource realizing this link port.
cpInstanceId	IdentifierInVnf	0..1	Identifier of the external CP of the VNF connected to this link port. There shall be at most one link port associated with any external connection point instance. The value refers to an "extCpInfo" item in the VnfInstance.

5.5.3.12a   Type: ExtLinkPortData

This type represents an externally provided link port to be used to connect an external connection point to an external VL. It shall comply with the provisions defined in table 5.5.3.12a-1.

Table 5.5.3.12a-1: Definition of the ExtLinkPortData data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this link port as provided by the entity that has created the link port.
resourceHandle	ResourceHandle	1	Reference to the virtualised resource realizing this link port.

5.5.3.13   Type: ResourceHandle

This type represents the information that allows addressing a virtualised resource that is used by a VNF instance. Information about the resource is available from the VIM. The ResourceHandle type shall comply with the provisions defined in table 5.5.3.13-1.

Table 5.5.3.13-1: Definition of the ResourceHandle data type

Attribute name	Data type	Cardinality	Description
vimConnectionId	Identifier	0..1	Identifier of the VIM connection to manage the resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable. See note 1.
resourceProviderId	Identifier	0..1	Identifier of the entity responsible for the management of the resource. This attribute shall only be supported and present when VNF-related resource management in indirect mode is applicable. The identification scheme is outside the scope of the present document.
resourceId	IdentifierInVim	1	Identifier of the resource in the scope of the VIM or the resource provider.
vimLevelResourceType	String	0..1	Type of the resource in the scope of the VIM or the resource provider. See note 2.
NOTE 1: The information about the VIM connection referenced by the VIM connection id is known to the VNFM. Moreover, the identifier of the VIM connection provides scope to the resourceId.			
NOTE 2: The value set of the "vimLevelResourceType" attribute is within the scope of the VIM or the resource provider and can be used as information that complements the ResourceHandle.			

5.5.3.14   VOID

5.5.3.15   VOID

5.5.3.15a   Type: CpProtocolInfo

This type describes the protocol layer(s) that a CP uses together with protocol-related information, like addresses. It shall comply with the provisions defined in table 5.5.3.15a-1.

Table 5.5.3.15a-1: Definition of the CpProtocolInfo data type

Attribute name	Data type	Cardinality	Description
layerProtocol	Enum (inlined)	1	The identifier of layer(s) and protocol(s) associated to the network address information. Permitted values: IP_OVER_ETHERNET. See note.
ipOverEthernet	IpOverEthernetAddressInfo	0..1	IP addresses over Ethernet to assign to the extCP instance. Shall be present if layerProtocol is equal to " IP_OVER_ETHERNET", and shall be absent otherwise.
NOTE: This attribute allows to signal the addition of further types of layer and protocol in future versions of the present document in a backwards-compatible way. In the current version of the present document, only IP over Ethernet is supported.			

5.5.3.16   Type: IpOverEthernetAddressInfo

This type represents information about a network address that has been assigned. It shall comply with the provisions defined in table 5.5.3.16-1.

Table 5.5.3.16-1: Definition of the IpOverEthernetAddressInfo data type

Attribute name	Data type	Cardinality	Description
macAddress	MacAddress	1	Assigned MAC address.
ipAddresses	Structure (inlined)	0..N	Addresses assigned to the extCP instance. Each entry represents IP addresses assigned by fixed or dynamic IP address assignment per subnet.
>type	Enum (inlined)	1	The type of the IP addresses. Permitted values: IPV4, IPV6.
>addresses	IpAddress	0..N	Fixed addresses assigned (from the subnet defined by "subnetId" if provided). See note.
>isDynamic	Boolean	0..1	Indicates whether this set of addresses was assigned dynamically (true) or based on address information provided as input from the API consumer (false). Shall be present if "addresses" is present and shall be absent otherwise.
>addressRange	Structure (inlined)	0..1	An IP address range used, e.g. in case of egress connections. See note.
>>minAddress	IpAddress	1	Lowest IP address belonging to the range
>>maxAddress	IpAddress	1	Highest IP address belonging to the range
>subnetId	IdentifierInVim	0..1	Subnet defined by the identifier of the subnet resource in the VIM. In case this attribute is present, IP addresses are bound to that subnet.
NOTE: Exactly one of "addresses" or "addressRange" shall be present.			

5.5.3.17   Type: MonitoringParameter

This type represents a monitoring parameter that is tracked by the VNFM, e.g. for auto-scaling purposes. It shall comply with the provisions defined in table 5.5.3.17-1.

Valid monitoring parameters of a VNF are defined in the VNFD.

> NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.

Table 5.5.3.17-1: Definition of the MonitoringParameter data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of the monitoring parameter defined in the VNFD.
name	String	0..1	Human readable name of the monitoring parameter, as defined in the VNFD.
value	(any type)	1	Value of the monitoring parameter known to the VNFM (e.g. obtained for autoscaling purposes). See note.
timeStamp	DateTime	1	Represents the point in time when the measurement has been performed, as known to the VNFM.
NOTE: The type of the "value" attribute (i.e. scalar, structure (Object in JSON) or array (of scalars, arrays or structures/Objects) is assumed to be defined in an external measurement specification.			

5.5.3.18   Type: LifecycleChangeNotificationsFilter

This type represents a subscription filter related to notifications about VNF lifecycle changes. It shall comply with the provisions defined in table 5.5.3.18-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 5.5.3.18-1: Definition of the LifecycleChangeNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscription Filter	VnfInstanceSubscription Filter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: • VnfLcmOperationOccurrenceNotification • VnfIdentifierCreationNotification • VnfIdentifierDeletionNotification See note.
operationTypes	LcmOperationType	0..N	Match particular VNF lifecycle operation types for the notification of type VnfLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "VnfLcmOperationOccurrenceNotification", and shall be absent otherwise.
operationStates	LcmOperationStateType	0..N	Match particular LCM operation state values as reported in notifications of type VnfLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "VnfLcmOperationOccurrenceNotification", and shall be absent otherwise.
NOTE: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.			

5.5.3.19   Type: AffectedVnfc

This type provides information about added, deleted, modified and temporary VNFCs. It shall comply with the provisions in table 5.5.3.19-1.

Table 5.5.3.19-1: Definition of the AffectedVnfc data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the Vnfc instance, identifying the applicable "vnfcResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
vduId	IdentifierInVnfd	1	Identifier of the related VDU in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change. Permitted values: • ADDED • REMOVED • MODIFIED • TEMPORARY For a temporary resource, an AffectedVnfc structure exists as long as the temporary resource exists.
computeResource	ResourceHandle	1	Reference to the VirtualCompute resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VnfcResourceInfo structure.
affectedVnfcCpIds	IdentifierInVnf	0..N	Identifiers of CP(s) of the VNFC instance that were affected by the change.
addedStorageResourceIds	IdentifierInVnf	0..N	References to VirtualStorage resources that have been added. Each value refers to a VirtualStorageResourceInfo item in the VnfInstance that was added to the VNFC. It shall be provided if at least one storage resource was added to the VNFC.
removedStorageResourceIds	IdentifierInVnf	0..N	References to VirtualStorage resources that have been removed. The value contains the identifier of a VirtualStorageResourceInfo item that has been removed from the VNFC, and might no longer exist in the VnfInstance. It shall be provided if at least one storage resource was removed from the VNFC.

5.5.3.20   Type: AffectedVirtualLink

This type provides information about added, deleted, modified and temporary VLs. It shall comply with the provisions in table 5.5.3.20-1.

Table 5.5.3.20-1: Definition of the AffectedVirtualLink data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the virtual link instance, identifying the applicable "vnfVirtualLinkResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
virtualLinkDescId	IdentifierInVnfd	1	Identifier of the related VLD in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change. Permitted values: • ADDED • REMOVED • MODIFIED • TEMPORARY • LINK_PORT_ADDED • LINK_PORT_REMOVED For a temporary resource, an AffectedVirtualLink structure exists as long as the temporary resource exists.
networkResource	ResourceHandle	1	Reference to the VirtualNetwork resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VnfVirtualLinkResourceInfo structure.

5.5.3.21   Type: AffectedVirtualStorage

This type provides information about added, deleted, modified and temporary virtual storage resources. It shall comply with the provisions in table 5.5.3.21-1.

Table 5.5.3.21-1: Definition of the AffectedVirtualStorage data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the storage instance, identifying the applicable "virtualStorageResourceInfo" entry in the "VnfInstance" data type (see clause 5.5.2.2).
virtualStorageDescId	IdentifierInVnfd	1	Identifier of the related VirtualStorage descriptor in the VNFD.
changeType	Enum (inlined)	1	Signals the type of change. Permitted values: • ADDED • REMOVED • MODIFIED • TEMPORARY For a temporary resource, an AffectedVirtualStorage structure exists as long as the temporary resource exists.
storageResource	ResourceHandle	1	Reference to the VirtualStorage resource. Detailed information is (for new and modified resources) or has been (for removed resources) available from the VIM.
metadata	KeyValuePairs	0..1	Metadata about this resource. The content of this attribute shall be a copy of the content of the "metadata" attribute of the VirtualStorageResourceInfo structure.

5.5.3.22   Type: LccnLinks

This type represents the links to resources that a notification can contain. It shall comply with the provisions defined in table 5.5.3.22-1.

Table 5.5.3.22-1: Definition of the LccnLinks data type

Attribute name	Data type	Cardinality	Description
vnfInstance	Link	1	Link to the resource representing the VNF instance to which the notified change applies.
subscription	Link	1	Link to the related subscription.
vnfLcmOpOcc	Link	0..1	Link to the VNF lifecycle management operation occurrence that this notification is related to. Shall be present if there is a related lifecycle operation occurrence.

5.5.3.23   Type: VnfcInfo

This type represents the information about a VNFC instance that is part of a VNF instance. It shall comply with the provisions defined in table 5.5.3.23-1.

Table 5.5.3.23-1: Definition of the VnfcInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the VNFC instance.
vduId	IdentifierInVnfd	1	Reference to the applicable VDU information element in the VNFD.
vnfcState	Enum (inlined)	1	State of the VNFC instance. Permitted values: • STARTED: The VNFC instance is up and running. • STOPPED: The VNFC instance has been shut down.
vnfcConfigurableProperties	KeyValuePairs	0..1	Current values of the configurable properties of the VNFC instance. Configurable properties referred in this attribute are declared in the VNFD. This attribute can be modified with the PATCH method.

5.5.3.24   Type: VnfcInfoModifications

This type represents modifications of an entry in an array of "VnfcInfo" objects. It shall comply with the provisions defined in table 5.5.3.24-1.

Table 5.5.3.24-1: Definition of the VnfcInfoModifications data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnf	1	Identifier of the VNFC instance of which the information is to be modified.
vnfcConfigurableProperties	KeyValuePairs	1	Changes of the configurable properties of the VNFC instance.

5.5.4   Referenced simple data types and enumerations

5.5.4.1   Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

5.5.4.2   Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

5.5.4.3   Enumeration: VnfOperationalStateType

The enumeration VnfOperationalStateType shall comply with the provisions defined in table 5.5.4.3-1.

Table 5.5.4.3-1: Enumeration VnfOperationalStateType

Enumeration value	Description
STARTED	The VNF instance is up and running.
STOPPED	The VNF instance has been shut down.

5.5.4.4   VOID

5.5.4.5   Enumeration: LcmOperationType

The enumeration LcmOpType represents those lifecycle operations that trigger a VNF lifecycle management operation occurrence notification. It shall comply with the provisions defined in table 5.5.4.5-1.

Table 5.5.4.5-1: Enumeration LcmOperationType

Enumeration value	Description
INSTANTIATE	Represents the "Instantiate VNF" LCM operation.
SCALE	Represents the "Scale VNF" LCM operation.
SCALE_TO_LEVEL	Represents the "Scale VNF to Level" LCM operation.
CHANGE_FLAVOUR	Represents the "Change VNF Flavour" LCM operation.
TERMINATE	Represents the "Terminate VNF" LCM operation.
HEAL	Represents the "Heal VNF" LCM operation.
OPERATE	Represents the "Operate VNF" LCM operation.
CHANGE_EXT_CONN	Represents the "Change external VNF connectivity" LCM operation.
MODIFY_INFO	Represents the "Modify VNF Information" LCM operation.

5.5.4.6   Enumeration: LcmOperationStateType

The enumeration LcmOperationStateType shall comply with the provisions defined in table 5.5.4.6-1. More information of the meaning of the states can be found in clause 5.6.2.2.

Table 5.5.4.6-1: Enumeration LcmOperationStateType

Enumeration value	Description
STARTING	The LCM operation is starting.
PROCESSING	The LCM operation is currently in execution.
COMPLETED	The LCM operation has been completed successfully.
FAILED_TEMP	The LCM operation has failed and execution has stopped, but the execution of the operation is not considered to be closed.
FAILED	The LCM operation has failed and it cannot be retried or rolled back, as it is determined that such action will not succeed.
ROLLING_BACK	The LCM operation is currently being rolled back.
ROLLED_BACK	The LCM operation has been successfully rolled back, i.e. The state of the VNF prior to the original operation invocation has been restored as closely as possible.

5.5.4.7   Enumeration: CancelModeType

The enumeration CancelModeType defines the valid modes of cancelling a VNF LCM operation occurrence. It shall comply with the provisions defined in table 5.5.4.7-1.

Table 5.5.4.7-1: Enumeration CancelModeType

Enumeration value	Description
GRACEFUL	If the VNF LCM operation occurrence is in "PROCESSING" or "ROLLING_BACK" state, the VNFM shall not start any new resource management operation and shall wait for the ongoing resource management operations in the underlying system, typically the VIM, to finish execution or to time out. After that, the VNFM shall put the operation occurrence into the FAILED_TEMP state. If the VNF LCM operation occurrence is in "STARTING" state, the VNFM shall not start any resource management operation and shall wait for the granting request to finish execution or time out. After that, the VNFM shall put the operation occurrence into the ROLLED_BACK state.
FORCEFUL	If the VNF LCM operation occurrence is in "PROCESSING" or "ROLLING_BACK" state, the VNFM shall not start any new resource management operation, shall cancel the ongoing resource management operations in the underlying system, typically the VIM, and shall wait for the cancellation to finish or to time out. After that, the VNFM shall put the operation occurrence into the FAILED_TEMP state. If the VNF LCM operation occurrence is in "STARTING" state, the VNFM shall not start any resource management operation and put the operation occurrence into the ROLLED_BACK state.

5.6 Handling of errors during VNF lifecycle management operations

5.6.1 Basic concepts (informative)

5.6.1.1 Motivation

VNF lifecycle management operation occurrences can fail. Failure can be caused by multiple reasons, which generally fall into the following categories:

- Transient errors which do not require intervention from a human operator or a higher-layer management entity for resolution, e.g. momentary network outage.

- "Permanent" errors which require such intervention.

It is unreasonable to expect that all errors can be resolved automatically, therefore the possibility of intervention will usually be incorporated in the system design as acknowledged means of error resolution.

5.6.1.2 Failure resolution strategies: Retry and Rollback

Most transient errors are handled best with a retry mechanism. Retry might happen automatically at the point of failure within the same LCM workflow (where it makes sense to limit the number of automatic retries). It is important to strive for designing retry operations that have no unintended side effects from the original invocation of the operation. This is called *idempotent retry*. Idempotent retry can also be used as an on-demand error resolution mechanism (see below) if the original operation failed because of a condition that has been resolved manually by the human operator or by a higher-level management entity, so idempotent retry is suitable for general error resolution in most cases.

However, even if a system is designed with idempotent retry capabilities, eventual success of the operation cannot be guaranteed. In this case, the resolution of the inconsistent state can be attempted by requesting to roll back the changes made by the operation. Therefore, rollback as an error handling strategy is also desired to be allowed in the system design.

In many cases, idempotent retry can resolve transient errors and lead to success eventually. Depending on the situation, rollback followed by a repetition of the operation could take longer than a successful retry, as rollback first removes allocated resources and then the repetition of the operation allocates them again, which costs time.

Therefore, it often makes sense to perform first idempotent retry, which is followed by rollback if the retry has failed. Idempotent retry is meaningful and useful for all operation types, but for some operations rollback is better suited and has a better chance of success. In general, rollback is well-suited for additive operations such as InstantiateVnf or scale out, while ill-suited for subtractive ones such as scale in or TerminateVnf, or for HealVnf.

Both rollback and idempotent retry can fail. In that case, the system can be left in an inconsistent state after a failed operation, which requires resolution by a higher level entity such as EM, NFVO or human operator.

5.6.1.3   Error handling at VNFM and EM

If the VNFM executes an LCM workflow and encounters a problem, the following options are possible:

- Stop on first error:

    - Once the VNFM encounters an error, the normal execution of the LCM workflow is interrupted, and an error handling procedure is triggered (automatic retry, automatic rollback, automatic fail, escalate). See the paragraphs below for description of error handling procedures.

    - It is assumed that all VNFs and all VNFMs support "stop on first error".

EXAMPLE 1:   EM is attempting to instantiate a VNF with 100 VNFCs. The first 97 VNFCs are instantiated successfully, however, an error occurs when attempting to instantiate VNFC #98. The VNFM stops execution and chooses which of the error handling options it invokes (note that it even could try multiple options after each other).

- Best Effort:

    - Each time the VNFM encounters an error, it is decided whether the execution of a part or all of the remaining steps of the LCM workflow is performed, or whether the execution is interrupted and an error handling procedure is triggered (automatic retry, automatic rollback, automatic fail, escalate). See the paragraphs below for description of error handling procedures.

    - Support of "best effort" requires a suitable workflow design.

    - It is therefore assumed that not all VNFs and not all VNFMs support "best effort".

EXAMPLE 2:   Same example as above. After the error occurs attempting to instantiate VNFC #98, the VNFM continues by creating #99 and #100, and then chooses which error handling options it invokes.

The VNFM has the following error handling procedures to react to errors (see clause 5.6.1.2 for general elaboration regarding retry and rollback):

- Automatic Retry: The VNFM retries (once or more) to continue the execution of the workflow without involving an external entity. Automatic retry of failed parts of the workflow might even be built into the workflow itself. Retry can eventually succeed or fail. Successful retry leads to the LCM operation to be reported as successful. Failed retry is typically escalated.

- Automatic Rollback: The VNFM rolls back the VNF to the state prior to starting the LCM operation without involving an external entity. Rollback can eventually succeed or can fail, preventing the VNF from reaching that previous state. Successful rollback leads to the LCM operation to be reported as rolled back. Failed rollback is typically escalated.

- Escalate: After failed automatic retry/retries, automatic rollback is typically not the first option in most situations, but the error is preferably reported to the EM for further resolution. The same applies if no automatic error resolution was attempted by the VNFM, or if automated rollback has failed. This is done by sending a VNF LCM operation occurrence notification.

- Unresolvable Error: The VNFM determines that the operation has failed and definitely cannot be recovered (e.g. if no retry and no rollback is possible), and that escalating the error will have no chance to lead to a resolution either. In this case, the VNFM would report that the operation has terminally failed. After that, other means of resolution can be attempted, such as the invocation of HealVnf, or manual procedures using the GUI of the VNFM or VIM to release stranded resources.

The EM has the following error handling procedures to react to error reports from the VNFM:

- On-demand retry: After the VNFM has reported the error to the EM, the EM or the human operator takes steps to resolve the situation that has led to the occurrence of the error. Subsequently, the retry of the operation is triggered towards the VNFM by the EM via the VNF LCM interface.

- On-demand rollback: After the VNFM has reported the error to the EM, and after the EM or the human operator has decided to roll back the operation, the rollback of the operation is triggered towards the VNFM by the EM via the VNF LCM interface.

- Fail: After the VNFM has reported the error to the EM, and after the EM or the human operator has determined that neither on-demand retry nor on-demand rollback will fix the error, the LCM operation can be declared as terminally failed towards the VNFM via the VNF LCM interface. After that, other means of resolution can be attempted, such as the invocation of HealVnf, or manual procedures using the GUI of the VNFM or VIM to release stranded resources.

NOTE: Error handling by the EM can involve escalations to the OSS/BSS, or to the NFVO via the OSS/BSS.

5.6.2   States and state transitions of a VNF lifecycle management operation occurrence

5.6.2.1   General

A VNF lifecycle management operation occurrence supports a number of states and error handling operations. The states and state transitions that shall be supported are shown in figure 5.6.2.1-1. Transitions labelled with underlined text represent error handling operations; other transitions represent conditions.
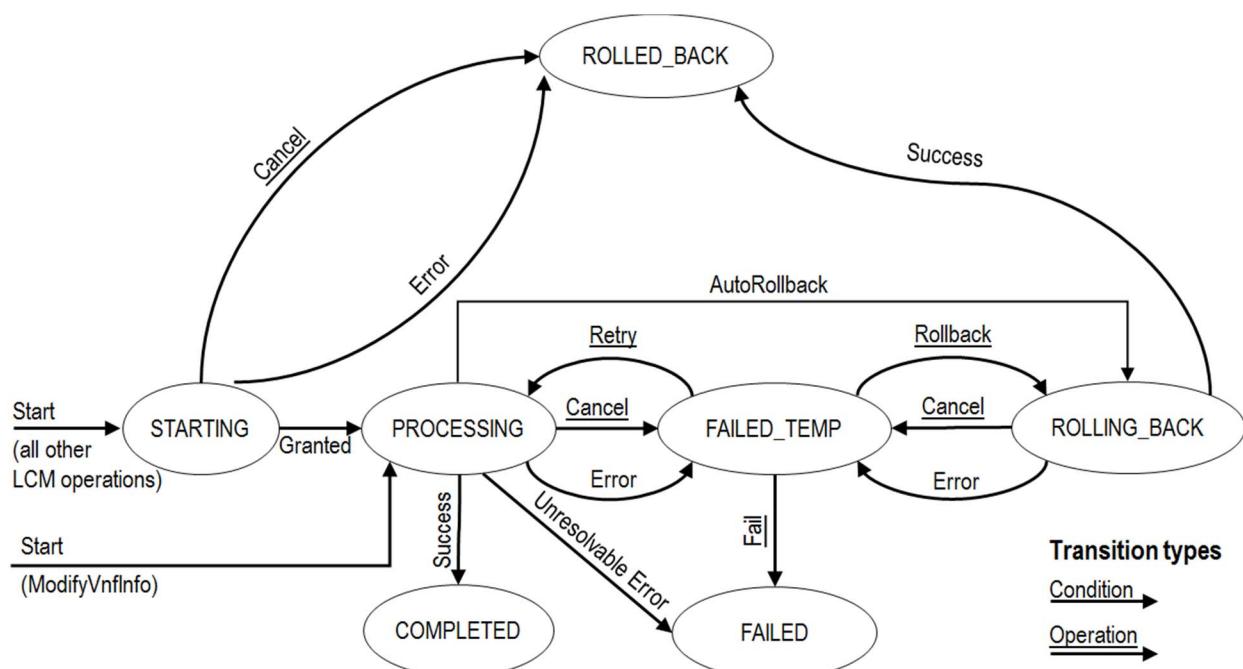
Figure 5.6.2.1-1: States of a VNF lifecycle management operation occurrence

5.6.2.2   States of a VNF lifecycle management operation occurrence

At each time, a VNF lifecycle management operation occurrence is in one of the following states. There are transitional states (states from which a different state can be reached) and terminal states (states from which no other state can be reached; i.e. the state of a VNF lifecycle management operation occurrence in a terminal state cannot change anymore).

STARTING: The operation is starting. This state represents the preparation phase of the operation, including invoking Grant Lifecycle Operation. This state has the following characteristics:

- This is the initial state for any LCM operation except ModifyVnfInformation.

- This is a transient state.

- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).

- In this state, the VNF lifecycle management operation occurrence does not perform any changes to the VNF instance or to resources.

- Once the VNF lifecycle operation has been granted, the VNF lifecycle management operation occurrence transits into the PROCESSING state.

- If the LCM operation is cancelled in the "STARTING" state, the VNF lifecycle management operation occurrence shall transit to the "ROLLED_BACK" state. The EM/VNF shall be prepared to receive the notification about the cancellation of the operation before and after having provided the grant. This is necessary to address possible race conditions.

- If an error occurs before the VNFM receives the grant response, or the grant is rejected, as no changes to the underlying VNF or resources were done, the VNF lifecycle management operation occurrence shall transit into the "ROLLED_BACK" state.

COMPLETED: The operation has completed successfully. This is a terminal state.

FAILED_TEMP: The operation has failed and execution has stopped, but the execution of the operation is not considered to be closed. This state has the following characteristics:

- This is a transient state.

- The grant received for the operation is still valid, and the granted resource changes are still foreseen for the VNF.

- This state may block other LCM operations from being executed on the same VNF instance (enforced by the VNFM, and up to VNF and VNFM capabilities).

- Retry and/or rollback and/or fail may be invoked for the operation.

- If the VNF LCM operation is retried, the VNF lifecycle management operation occurrence shall transit into the "PROCESSING" state.

- If the VNF LCM operation is rolled back, the VNF lifecycle management operation occurrence shall transit into the "ROLLING_BACK" state.

- If the VNF LCM operation is marked as "failed", the VNF lifecycle management operation occurrence shall transit into the "FAILED" state.

- Operation cancellation and failure to roll back should result in FAILED_TEMP.

FAILED: The operation has failed and it cannot be retried or rolled back, as it is determined that such action will not succeed. This state has the following characteristics:

- This is a terminal state.

- Such an operation state is typically the result of a decision of a higher layer management entity (EM/VNF) or its human operator that an operation in FAILED_TEMP state cannot be retried or rolled back ("Fail").

- Such an operation state can also be reached immediately in case of failure of an operation in "PROCESSING" state that can neither be retried nor rolled back ("Unresolvable Error").

- The result of the LCM operation (the actual resource changes) can show an inconsistent state of the VNF, and can reflect partial resource changes compared to the granted changes. Nevertheless, these changes shall be synchronized between the VNFM and EM/VNF (by reporting them in the LCCN, and by allowing the EM/VNF to obtain them on request) in order for other VNF LCM operations (e.g. Heal, Terminate) to be guaranteed to work on resources that are known to the EM/VNF.

- The fact that a LCM operation is in "FAILED" state shall not block other operations from execution on the VNF instance by the VNFM. However, the VNF instance may itself be in a state that disallows certain operations.

ROLLED_BACK: The state of the VNF prior to the original operation invocation has been restored as closely as possible. This state has the following characteristics:

- This is a terminal state.

- This may involve recreating some resources that have been deleted by the operation, the recreated resources should be as similar as possible to the deleted ones. Differences between original resources and re-created ones may include a different resource identity, but also different dynamic attributes such as an IP address.

PROCESSING: The LCM operation is currently in execution. This state has the following characteristics:

- This is the initial state for the ModifyVnfInformation operation.

- This is a transient state.

- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).

- The operations "Retry" and "Rollback" shall not be permitted to be invoked for an operation that is in this state.

- All failures of procedures executed by the VNFM as part of the LCM operation while in "PROCESSING" state should result by default in transiting to FAILED_TEMP, with the following two alternative options:

  - If a failure occurs in the "PROCESSING" state from which the VNFM knows that the VNF instance can be brought into a consistent state by immediately rolling back the operation, the VNF lifecycle management operation occurrence may transit directly into the "ROLLING_BACK" state ("AutoRollback"). For the "ModifyVnfInformation" operation, AutoRollback is the typical error handling method.

  - If a failure occurs in the "PROCESSING" state from which the VNFM knows that it can neither be fixed by retrying nor be rolled back, the VNF lifecycle management operation occurrence may transit directly into the "FAILED" state ("Unresolvable Error").

- If a "cancel" request was issued during the operation is in "PROCESSING" state, processing will be cancelled but this might not be immediate. This is represented by a flag in the data model that indicates there is a pending "cancel" request for this state. Upon successful cancellation, the VNF lifecycle management operation occurrence shall transit into the "FAILED_TEMP" state.

ROLLING_BACK: The LCM operation is currently being rolled back. This state has the following characteristics:

- This is a transient state.

- This state may block other LCM operations from being executed on the same VNF instance (up to VNF and VNFM implementation).

- The operations "Retry" and "Rollback" shall not be permitted to be invoked for an operation that is in this state.

- If a "Cancel" request was issued during the operation is in "ROLLING_BACK" state, rolling back will be cancelled but this might not be immediate. This is represented by a flag in the data model that indicates there is a pending "Cancel" request for this state. Upon successful cancellation, the VNF lifecycle management operation occurrence shall transit into the "FAILED_TEMP" state.

- If a failure occurs during rolling back, the operation should transition to the "FAILED_TEMP" state.

- Upon successful rollback, the VNF lifecycle management operation occurrence shall transit into the "ROLLED_BACK" state.

In addition, the following provisions apply to VNF lifecycle management operation occurrence notifications:

- The "start" notification (i.e. notificationStatus="START") shall be sent when the operation enters one of states "STARTING", "PROCESSING" and "ROLLING_BACK" from another state, indicating the state entered.

- The "result" notification (i.e. notificationStatus="RESULT") shall be sent when the VNF LCM operation occurrence enters one of the error states "FAILED_TEMP", "FAILED", "ROLLED_BACK", indicating the state entered, the error cause and the changes to the VNF's resources since the operation was initially started.

- The "result" notification (i.e. notificationStatus="RESULT") shall be sent when the operation enters the success state "COMPLETED", indicating the state entered and the changes to the VNF's resources.

Such a notification scheme allows the EM/VNF to keep in sync with changes to the VNF's resources by an ongoing LCM operation. If the notification relates to a transient state, further changes can be expected. If the notification relates to a terminal state, no further changes to the VNF's resources will be performed by the related VNF lifecycle management operation occurrence, and the EM/VNF can use the information in the notification to synchronize its internal state with the result of the LCM operation. In case of loss of notifications, the EM/VNF can read the resource that represents the VNF lifecycle management operation occurrence to obtain the same information.

5.6.2.3   Error handling operations that change the state of a VNF lifecycle management operation occurrence

Retry: This operation retries a VNF lifecycle operation. It has the following characteristics:

- Execution of "Retry" for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.

- "Retry" shall operate within the bounds of the Grant for the LCM operation.

- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Rollback: This operation rolls back a VNF lifecycle operation. It has the following characteristics:

- Execution of "Rollback" for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.

- "Rollback" shall operate within the bounds of the Grant for the LCM operation, an additionally may execute the inverse of granted LCM operations (e.g. if a resource deletion was granted, rollback might re-create the deleted resource or a similar resource).

- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Fail: This operation transits the VNF lifecycle management operation occurrence into the terminal "FAILED" state. It has the following characteristics:

- Execution of "Fail" shall be supported for an LCM operation on a particular VNF if at least one of Retry, Rollback, Cancel is supported for this operation.

- The operation may be invoked via an interface, or the VNFM may invoke the operation per its own decision.

Cancel: This operation cancels an ongoing VNF lifecycle management operation, its Retry or Rollback. It has the following characteristics:

- Execution of Cancel for an actual LCM operation on a particular VNF may be supported, depending on characteristics of the VNF and the LCM operation.

- The "Cancel" operation need not have immediate effect, depending on the capabilities of the underlying systems, and the currently executed resource management operation.

- Two modes of cancellation are supported: graceful and forceful:

    - When executing the *graceful* "Cancel" operation, the VNFM will not initiate any new operation towards the underlying systems, will wait until the currently executed operations finish, fail or time out in the VNFM, and will then put the VNF lifecycle management operation occurrence into the "FAILED_TEMP" state.

-        When executing the *forceful* "Cancel" operation, the VNFM will cancel all ongoing operations in the underlying systems for which cancellation is supported, will not initiate any new operation towards the underlying systems, will wait for the requested cancellations to finish, fail or time out in the VNFM, and will then put the VNF lifecycle management operation occurrence into the "FAILED_TEMP" state.

NOTE: In both modes, the time-out is determined by means outside the scope of the present document.

-        In "STARTING" state, there is no difference between the graceful and the forceful cancellation mode.

- Executing "Cancel" can lead to inconsistencies between the information that the VNFM has about the state of the resources of the VNF, and their actual state. The probability of such inconsistencies is bigger when using the *forceful* cancellation mode.

5.6.3   Detailed flows

5.6.3.1   Immediate failure

If the VNF LCM operation fails immediately, i.e. it returns an HTTP error, then the operation has not started, and no "VNF LCM operation occurrence" resource has been created. Also, neither a "start" VNF lifecycle management operation occurrence notification nor a Grant request has been sent. The operation cannot be retried, but the same operation may be invoked again from the API. The VNF instance is not changed by a synchronous failure, so no special error handling is required.

Figure 5.6.3.1-1 illustrates the flow.
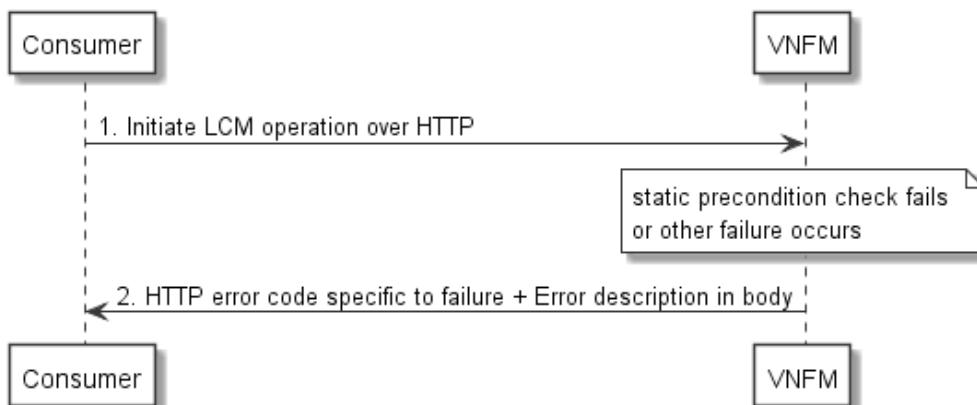
Figure 5.6.3.1-1: Immediate failure of a VNF LCM operation

5.6.3.2   Failure in "STARTING" state

This error scenario assumes that the "VNF LCM operation occurrence" resource has been created and the "start" VNF lifecycle management operation occurrence notification has been sent.

If the operation fails before the VNFM receives the Grant response, or the Grant is rejected, persistent change to the state of the VNF cannot have happened. Therefore, it is assumed that this operation enters the ROLLED_BACK state immediately. Figure 5.6.3.2-1 illustrates the flow.
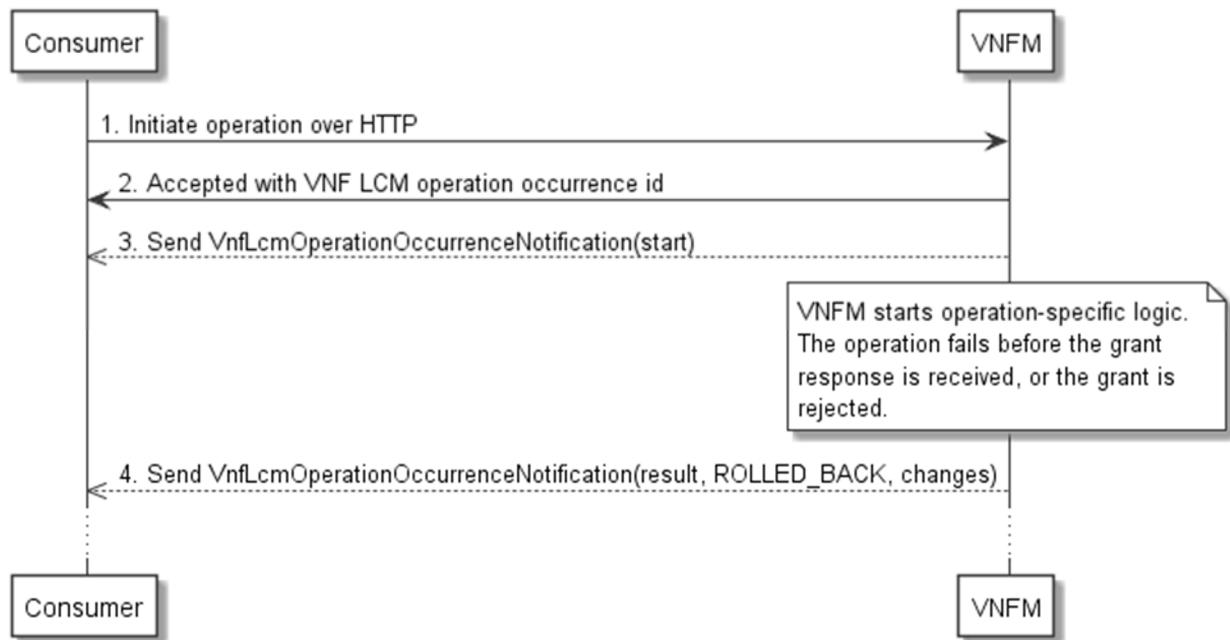
Figure 5.6.3.2-1: Failure of a VNF LCM operation before applying any change to the VNF instance

5.6.3.3   Failure during actual LCM operation execution

After a failed resource management operation, automatic retry can be invoked by the VNFM itself. These invocations are not visible outside of the VNFM, as the VNF LCM operation occurrence stays in "PROCESSING" state during these automatic retries. If these do not resolve the issue, intervention (typically by a human operator) is necessary. For that purpose, the LCM operation is set into a temporary failure state, and the EM is notified. The human operator performs a root cause analysis and eventually resolves the obstacle. Subsequently, and if supported, the operation can be retried, rolled-back or determined as permanently failed. Figure 5.6.3.3-1 illustrates the possible options.

  NOTE 1: Excluding automated rollback which is seen as a rare option.

  NOTE 2: Excluding "start" notifications (i.e. notificationStatus="START") for simplification purposes.
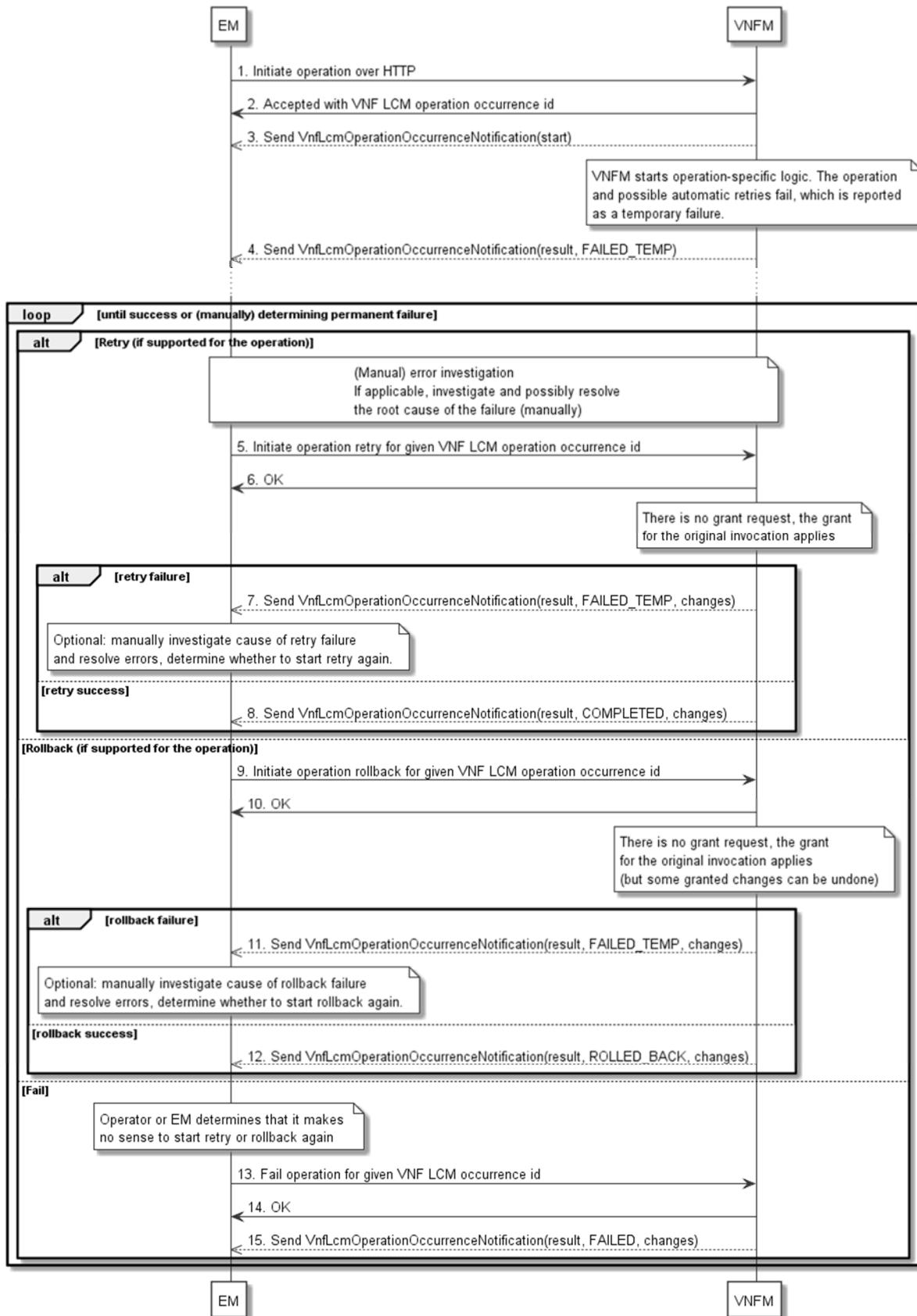
Figure 5.6.3.3-1: Handling failures during the actual execution of a VNF LCM operation

5.6.3.4 LCM operation cancellation

The cancellation of an LCM operation that is in PROCESSING or ROLLING_BACK state is handled like any other error that leads to stopping the execution of the VNF LCM workflow before it can be successfully completed. The VNF LCM operation transits into the FAILED_TEMP state which allows root cause analysis, possible fixing of the root cause, followed by retrying, rolling back, or finally failing of the operation.

The cancellation of an operation in STARTING state (i.e. until the Grant is received) transits the operation into the ROLLED_BACK state, as no changes to the resources or VNF instance have been performed.

6 VNF Performance Management interface

6.1 Description

This interface allows providing performance management (measurement results collection and notifications) related to VNFs. Performance information on a given VNF/VNFC instance results from performance information of the virtualised resources that is collected from the VIM and mapped to this VNF/VNFC instance. Collection and reporting of performance information is controlled by a PM job that groups details of performance collection and reporting information.

When new performance information is available, the consumer is notified using the notification PerformanceInformationAvailableNotification.

The operations provided through this interface which will be consumed by the EM are:

- Create PM Job

- Query PM Job

- Delete PM Job

- Create Threshold

- Query Threshold

- Delete Threshold

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

The operations provided through this interface which will be consumed by the VNF are:

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

6.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2.

The string "vnfpm" shall be used to represent {apiName}. The {apiVersion} shall be set to "v1" for the present document. All resource URIs in the clauses below are defined relative to the above base URI.

Figure 6.2-1 shows the overall resource URI structure defined for the performance management API.
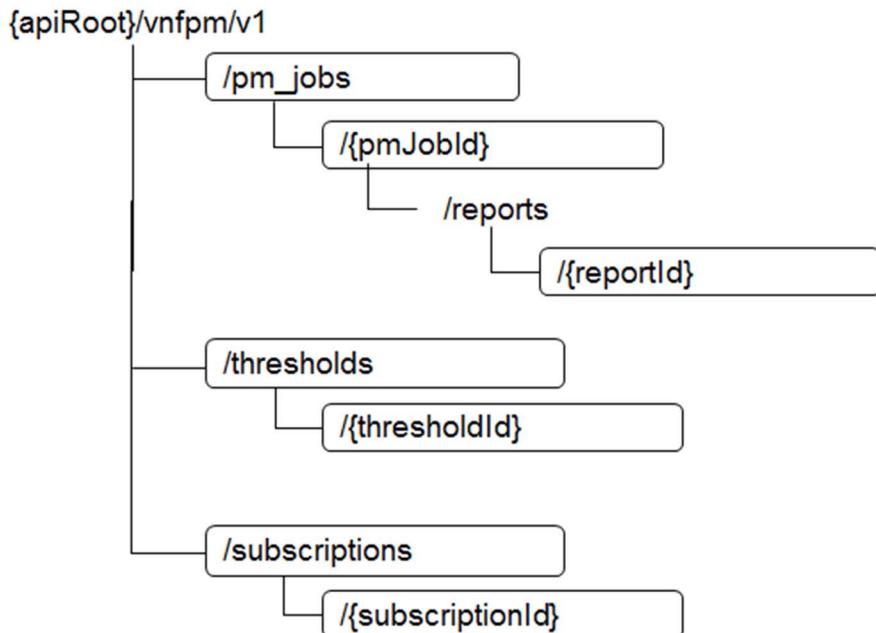
Figure 6.2-1: Resource URI structure of the VNF Performance Management interface

Table 6.2-1 lists the individual resources defined, and the applicable HTTP methods.

Table 6.2-1: Resources and methods overview of the VNF Performance Management interface

Resource name	Resource URI	HTTP Method	Meaning
PM jobs	/pm_jobs	POST	Create a PM job
		GET	Query PM jobs
Individual PM job	/pm_jobs/{pmJobId}	GET	Read a single PM job
		DELETE	Delete a PM job
Individual performance report	/pm_jobs/{pmJobId}/reports/{reportId}	GET	Read an individual performance report
Thresholds	/thresholds	POST	Create a threshold
		GET	Query thresholds
Individual threshold	/thresholds/{thresholdId}	GET	Query a single threshold
		DELETE	Delete a threshold
Subscriptions	/subscriptions	POST	Subscribe to PM notifications
		GET	Query PM related subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	Query a single PM related subscription
		DELETE	Terminate a subscription
Notification endpoint	(client-defined)	POST	Notify about PM related events
		GET	Test the notification endpoint

6.3 Sequence diagrams (informative)

6.3.1 Flow of creating a PM job

This clause describes a sequence for creating a performance management jobs.
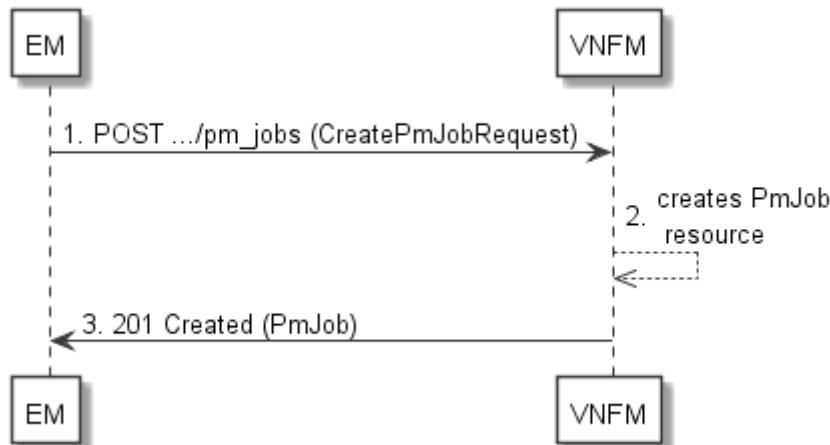
Figure 6.3.1-1: Flow of PM job creation

PM job creation, as illustrated in figure 6.3.1-1, consists of the following steps:

1) If the EM intends to create a PM job, it sends a POST request to the "PM jobs" resource, including one data structure of type "CreatePmJobRequest" in the payload body.

2) The VNFM creates a PM job instance.

3) The VNFM returns a "201 Created" response to the EM, and includes in the payload body a representation of the PM job just created.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.2 Flow of querying/reading PM jobs

This clause describes a sequence for querying/reading performance management jobs.
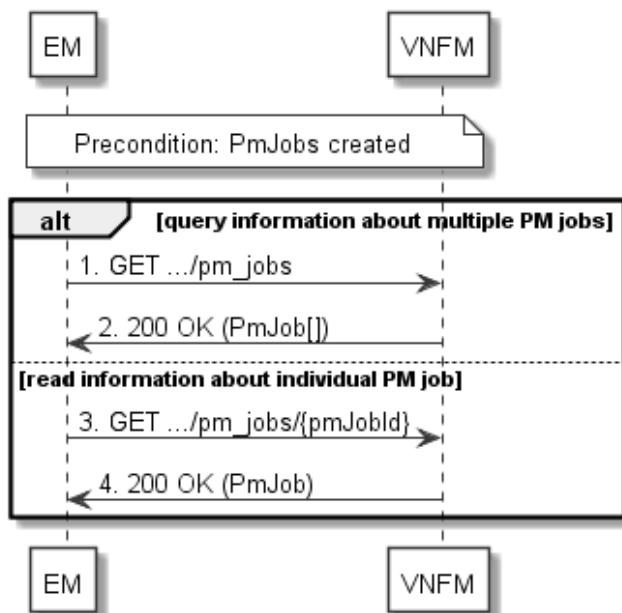
Figure 6.3.2-1: Flow of PM jobs query/read

PM jobs query/read, as illustrated in figure 6.3.2-1, consists of the following steps:

1) If the EM intends to query all PM jobs, it sends a GET request to the "PM jobs" resource.

2)   The VNFM returns a "200 OK" response to the EM, and includes zero or more data structures of type "PmJob" in the payload body.

3)   If the EM intends to read information about a particular PM job, it sends a GET request to the "Individual PM job" resource, addressed by the appropriate PM job identifier in its resource URI.

4)   The VNFM returns a "200 OK" response to the EM, and includes one data structure of type "PmJob" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.3   Flow of deleting a PM job

This clause describes a sequence for deleting a performance management jobs.
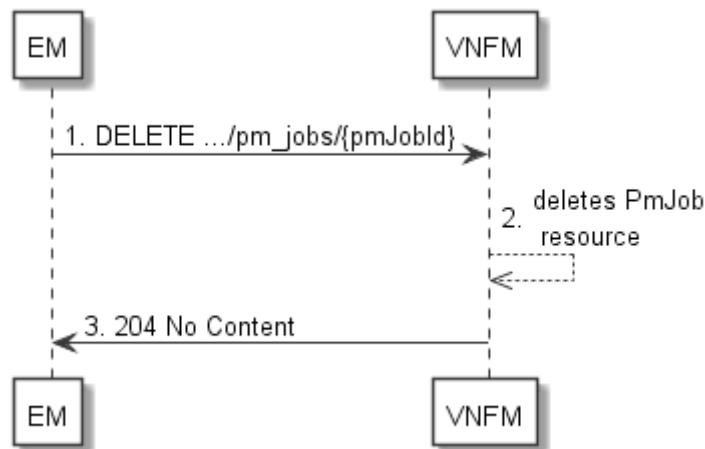
Figure 6.3.3-1: Flow of PM job deletion

PM job deletion, as illustrated in figure 6.3.3-1, consists of the following steps:

1)   If the EM intends to delete a PM job, it sends a DELETE request to the "Individual PM job" resource addressed by the appropriate PM job identifier in its resource URI.

2)   The VNFM returns a response with a "204 No Content" response code and an empty payload body to the EM.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.4   Flow of obtaining performance reports

This clause describes a sequence for obtaining performance reports.
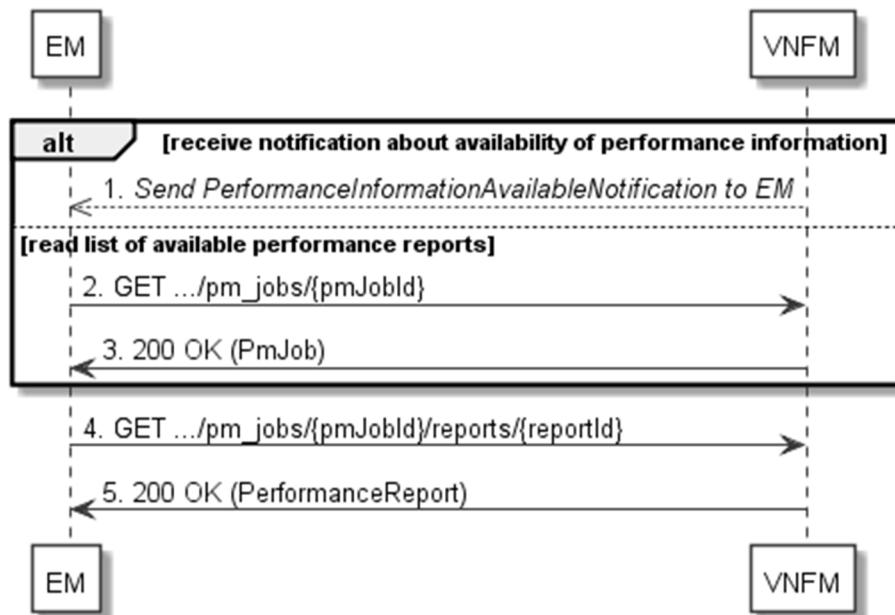
Figure 6.3.4-1: Flow of obtaining performance reports

Obtaining a performance report, as illustrated in figure 6.3.4-1, consists of the following steps:

1) The VNFM sends to the EM a PerformanceInformationAvailableNotification (see clause 6.3.9) that indicates the availability of a new performance report, including a link from which the report can be obtained.

2) Alternatively, the EM sends a GET request to the "Individual PM job" resource, to obtain a representation of the PM job resource including information about performance reports that are available for this PM job, including their URIs.

3) In that case, the VNFM returns a "200 OK" response to the EM, and includes a data structure of type "PmJob" in the payload body.

4) The EM sends to the VNFM a GET request to the URI obtained either in step (1) or step (3), in order to read a performance report resource.

5) The VNFM returns a "200 OK" response to the EM, and includes a data structure of type "PerformanceReport" in the payload body.

6.3.5 Flow of creating a threshold

This clause describes a sequence for creating a performance management threshold.
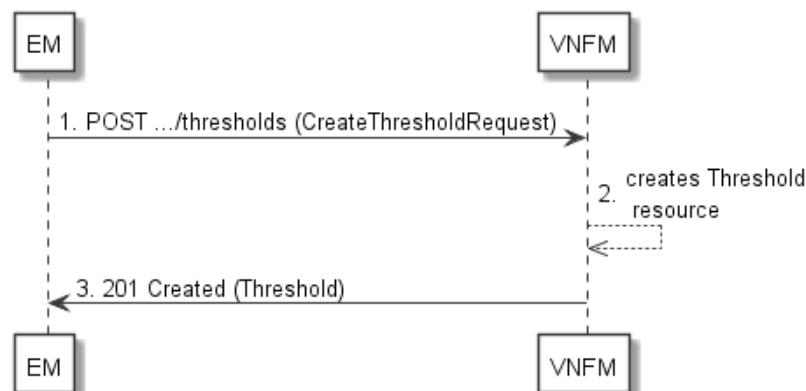
Figure 6.3.5-1: Flow of threshold creation

Threshold creation, as illustrated in figure 6.3.5-1, consists of the following steps:

1) If the EM intends to create a threshold, it sends a POST request to the "Thresholds" resource, including a data structure of type "CreateThresholdRequest" in the payload body.

2) The VNFM creates a threshold instance.

3) The VNFM returns a "201 Created" response to the EM, and includes in the payload body a representation of the threshold just created.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.6   Flow of querying/reading thresholds

This clause describes a sequence for querying/reading performance management thresholds.
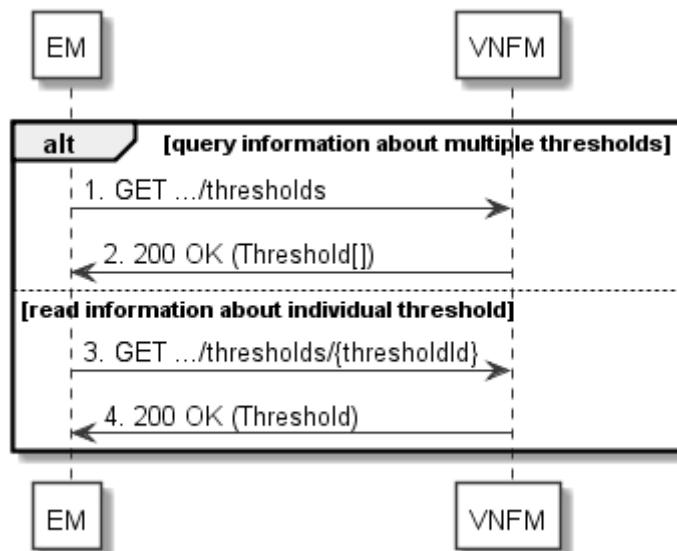
Figure 6.3.6-1: Flow of thresholds query/read

Threshold query/read, as illustrated in figure 6.3.6-1, consists of the following steps:

1) If the EM intends to query all thresholds, it sends a GET request to the "Thresholds" resource.

2) The VNFM returns a "200 OK" response to the EM, and includes zero or more data structures of type "Threshold" in the payload body.

3) If the EM intends to read information about a particular threshold, it sends a GET request to the "Individual threshold" resource, addressed by the appropriate threshold identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the EM, and includes a data structure of type "Threshold" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.7   Flow of deleting thresholds

This clause describes a sequence for deleting performance management thresholds.
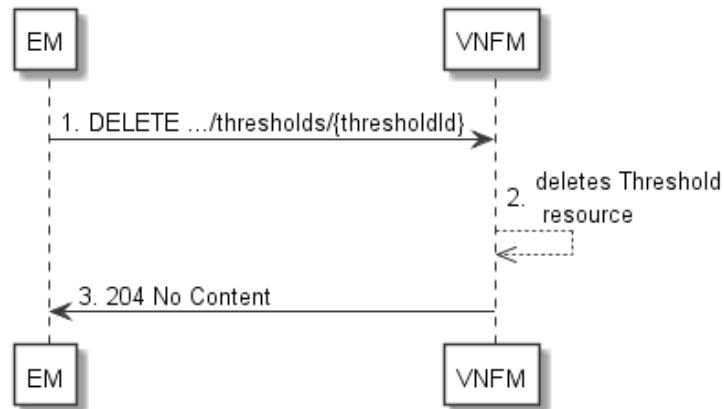
Figure 6.3.7-1: Flow of threshold deletion

Threshold deletion, as illustrated in figure 6.3.7-1, consists of the following steps:

1) If the EM intends to delete a particular threshold, it sends a DELETE request to the "Individual threshold" resource, addressed by the appropriate threshold identifier in its resource URI.

2) The VNFM returns a "204 No Content" response code to the EM. The response body shall be empty.

Error handling: In case of failure, appropriate error information is provided in the response.

6.3.8   Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF performance management between the EM and the VNFM.
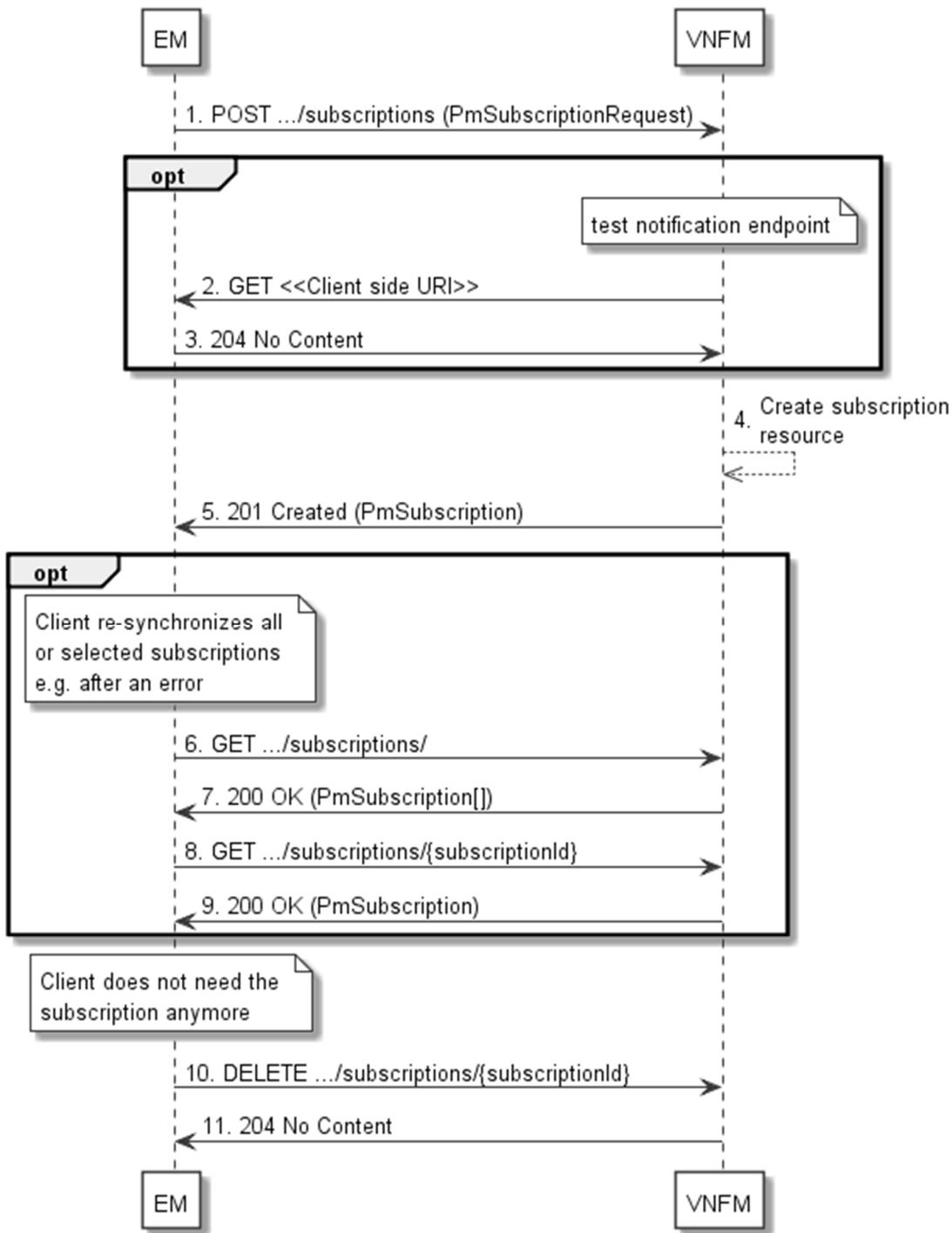
Figure 6.3.8-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 6.3.8-1:

1) The EM sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "PmSubscriptionRequest". This data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the EM as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3) In that case, the EM returns a "204 No Content" response to indicate success.

4) The VNFM creates a new subscription to notifications related to VNF performance management, and a resource that represents this subscription.

5) The VNFM returns a "201 Created" response containing a data structure of type "PmSubscription", representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6) Optionally, for example when trying to recover from an error situation, the EM may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the VNFM returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the EM.

8) Optionally, for example when trying to recover from an error situation, the EM may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10) When the EM does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription.

11) The VNFM acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

6.3.9   Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF performance management between the EM and the VNFM.
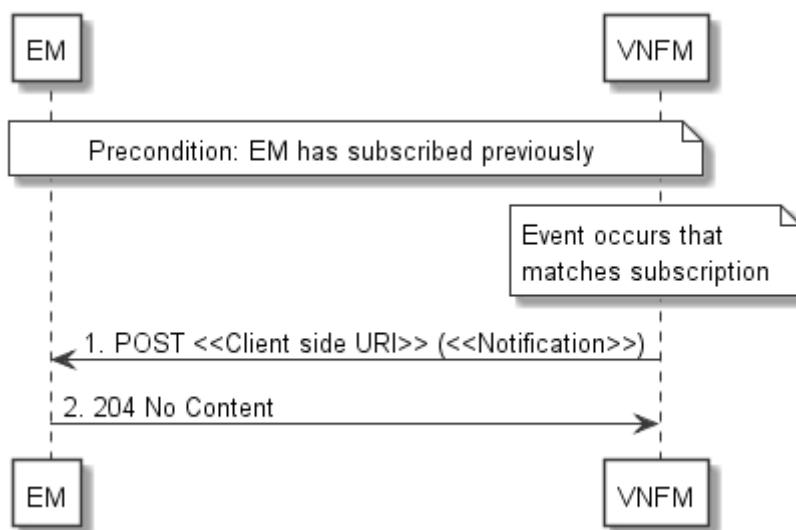
Figure 6.3.9-1: Flow of sending notifications

Precondition: The EM has subscribed previously for notifications related to VNF performance management.

The procedure consists of the following steps as illustrated in figure 6.3.9-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the EM has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API.

2) The EM acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the EM, it can retry sending the notification.

6.4 Resources

6.4.1 Introduction

This clause defines all the resources and methods provided by the performance management API.

6.4.2 Resource: PM jobs

6.4.2.1 Description

This resource represents PM jobs. The client can use this resource to create and query PM jobs.

6.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs

This resource shall support the resource URI variables defined in table 6.4.2.2-1.

Table 6.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.2.3 Resource methods

6.4.2.3.1 POST

The POST method creates a PM job.

This method shall follow the provisions specified in the tables 6.4.2.3.1-1 and 6.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.2.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description	
	CreatePmJobRequest	1		PM job creation request	
	Data type	**Cardinality**	**Response Codes**	**Description**	
Response body	PmJob	1	201 Created	The PM job was created successfully. The response body shall contain a representation of the created PM job resource, as defined in clause 6.5.2.7. The HTTP response shall include a "Location" HTTP header that points to the created PM job resource.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

6.4.2.3.2   GET

The client can use this method to retrieve information about PM jobs.

This method shall follow the provisions specified in the tables 6.4.2.3.2-1 and 6.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. All attribute names that appear in the PmJob and in data types referenced from it shall be supported in attribute-based filtering parameters.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The VNFM shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The VNFM should support this parameter.
exclude_default	0..1	Indicates to exclude the following complex attributes from the response. See clause 4.3.3 for details. The VNFM shall support this parameter. The following attributes shall be excluded from the PmJob structure in the response body if this parameter is provided, or none of the parameters "all_fields", "fields", "exclude_fields", "exclude_default" are provided: • reports

Table 6.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	PmJob	0..N	200 OK	Information about zero or more PM jobs was queried successfully. The response body shall contain representations of zero or more PM jobs, as defined in clause 6.5.2.7.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute selector. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.2.3.3   PUT

Not supported.

6.4.2.3.4 PATCH

Not supported.

6.4.2.3.5 DELETE

Not supported.

6.4.3 Resource: Individual PM job

6.4.3.1 Description

This resource represents an individual PM job. The client can use this resource to delete and read the underlying PM job.

6.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs/{pmJobId}

This resource shall support the resource URI variables defined in table 6.4.3.2-1.

Table 6.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
pmJobId	Identifier of the PM job. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new PM job resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

6.4.3.3 Resource methods

6.4.3.3.1 POST

Not supported.

6.4.3.3.2 GET

The client can use this method for reading an individual PM job.

This method shall follow the provisions specified in the tables 6.4.3.3.2-1 and 6.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmJob	1	200 OK	Information about an individual PM job was queried successfully. The response body shall contain a representation of the PM job resource, as defined in clause 6.5.2.7.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.3.3.3   PUT

Not supported.

6.4.3.3.4   PATCH

Not supported.

6.4.3.3.5   DELETE

This method terminates an individual PM job.

This method shall follow the provisions specified in the tables 6.4.3.3.5-1 and 6.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.3.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		204 No Content	The PM job was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.4   Resource: Individual performance report

6.4.4.1   Description

This resource represents an individual performance report that was collected by a PM job. The client can use this resource to read the performance report. The URI of this report can be obtained from a PerformanceInformationAvailableNotification (see clause 6.5.2.5) or from the representation of the "Individual PM job" resource.

It is determined by means outside the scope of the present document, such as configuration or policy, how long an individual performance report is available.

6.4.4.2   Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/pm_jobs/{pmJobId}/reports/{reportId}

This resource shall support the resource URI variables defined in table 6.4.4.2-1.

Table 6.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
pmJobId	Identifier of the PM job.
reportId	Identifier of the performance report.

6.4.4.3   Resource methods

6.4.4.3.1   POST

Not supported.

6.4.4.3.2   GET

The client can use this method for reading an individual performance report.

This method shall follow the provisions specified in the tables 6.4.4.3.2-1 and 6.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.4.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	PerformanceReport	1	200 OK	Information of an individual performance report was read successfully. The response body shall contain a representation of the performance report resource, as defined in clause 6.5.2.10.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.4.3.3   PUT

Not supported.

6.4.4.3.4   PATCH

Not supported.

6.4.4.3.5   DELETE

Not supported.

6.4.5   Resource: Thresholds

6.4.5.1   Description

This resource represents thresholds. The client can use this resource to create and query thresholds.

6.4.5.2   Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/thresholds

This resource shall support the resource URI variables defined in table 6.4.5.2-1.

Table 6.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.5.3   Resource methods

6.4.5.3.1   POST

The POST method can be used by the client to create a threshold.

This method shall follow the provisions specified in the tables 6.4.5.3.1-1 and 6.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.5.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description		
	CreateThresholdRequest	1	Request parameters to create a threshold resource.		
	Data type	**Cardinality**	**Response Codes**	**Description**	
Response body	Threshold	1	201 Created	A threshold was created successfully. The response body shall contain a representation of the created threshold resource, as defined in clause 6.5.2.9. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created threshold resource.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

6.4.5.3.2   GET

The client can use this method to query information about thresholds.

This method shall follow the provisions specified in the tables 6.4.5.3.2-1 and 6.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. All attribute names that appear in the Thresholds data type and in data types referenced from it shall be supported in attribute-based filtering parameters.

NOTE: There are no attribute selectors defined for this resource as the threshold attributes with cardinality 0..1 or 0..N are not structurally complex in nature.

Table 6.4.5.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	Threshold	0..N	200 OK	Information about zero or more thresholds was queried successfully. The response body shall contain representations of zero or more thresholds, as defined in clause 6.5.2.9.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.5.3.3   PUT

Not supported.

6.4.5.3.4   PATCH

Not supported.

6.4.5.3.5   DELETE

Not supported.

6.4.6   Resource: Individual threshold

6.4.6.1   Description

This resource represents an individual threshold.

6.4.6.2   Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/thresholds/{thresholdId}

This resource shall support the resource URI variables defined in table 6.4.6.2-1.

Table 6.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
thresholdId	Identifier of the threshold. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new threshold resource. It can also be retrieved from the "id" attribute in the payload body of that response.

6.4.6.3   Resource methods

6.4.6.3.1   POST

Not supported.

6.4.6.3.2   GET

The client can use this method for reading an individual threshold

This method shall follow the provisions specified in the tables 6.4.6.3.2-1 and 6.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.6.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	Threshold	1	200 OK	Information about an individual threshold was queried successfully. The response body shall contain a representation of the threshold, as defined in clause 6.5.2.9.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

6.4.6.3.3   PUT

Not supported.

6.4.6.3.4   PATCH

Not supported.

6.4.6.3.5   DELETE

This method allows to delete a threshold.

This method shall follow the provisions specified in the tables 6.4.6.3.5-1 and 6.4.6.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.6.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	n/a		204 No Content	The threshold was deleted successfully. The response body shall be empty.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

6.4.7   Resource: Subscriptions

6.4.7.1   Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF performance management and to query its subscriptions.

6.4.7.2   Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/subscriptions

This resource shall support the resource URI variables defined in table 6.4.7.2-1.

Table 6.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.7.3   Resource methods

6.4.7.3.1   POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 6.4.7.3.1-1 and 6.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the EM or VNF, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 6.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Description
	PmSubscriptionRequest	1		Details of the subscription to be created.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmSubscription	1	201 Created	The subscription was created successfully. A representation of the created subscription resource shall be returned in the response body, as defined in clause 6.5.2.3. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.7.3.2   GET

The client can use this method to query the list of active subscriptions to Performance management notifications subscribed by the client.

This method shall follow the provisions specified in the tables 6.4.7.3.2-1 and 6.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. The VNF may supply its instance Id as an attribute filter. All attribute names that appear in the PmSubscription and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 6.4.7.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain the representations of all active subscriptions of the functional block that invokes the method, as defined in clause 6.5.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.7.3.3   PUT

Not supported.

6.4.7.3.4   PATCH

Not supported.

6.4.7.3.5   DELETE

Not supported.

6.4.8   Resource: Individual subscription

6.4.8.1   Description

This resource represents an individual subscription for notifications about performance management related events. The client can use this resource to read and to terminate a subscription to notifications related to VNF performance management.

6.4.8.2   Resource definition

The resource URI is:

{apiRoot}/vnfpm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 6.4.8.2-1.

Table 6.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of the subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

6.4.8.3   Resource methods

6.4.8.3.1   POST

Not supported.

6.4.8.3.2   GET

The client can use this method for reading an individual subscription about Performance management notifications subscribed by the client.

This method shall follow the provisions specified in the tables 6.4.8.3.2-1 and 6.4.8.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.8.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**	
	PmSubscription	1	200 OK	The subscription was read successfully. The response body shall contain a representation of the subscription resource, as defined in clause 6.5.2.3.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

6.4.8.3.3   PUT

Not supported.

6.4.8.3.4   PATCH

Not supported.

6.4.8.3.5   DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the tables 6.4.8.3.5-1 and 6.4.8.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.8.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.8.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description			
	n/a					
Response body	Data type	Cardinality	Response Codes	Description		
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.		
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.		

6.4.9   Resource: Notification endpoint

6.4.9.1   Description

This resource represents a notification endpoint for VNF performance management.

The API producer can use this resource to send notifications related to performance management events to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

6.4.9.2   Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 6.4.9.2-1.

Table 6.4.9.2-1: Resource URI variables for this resource

Name	Definition
n/a	

6.4.9.3   Resource methods

6.4.9.3.1   POST

The POST method delivers a notification regarding a performance management event from the server to the client.

This method shall follow the provisions specified in the tables 6.4.9.3.1-1 and 6.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	PerformanceInformation AvailableNotification	1		Notification about performance information availability
	ThresholdCrossedNotific ation	1		Notification about threshold crossing
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		204 No Content	The notification was delivered successfully.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned. The VNFM shall not retry sending the notification if a 4xx/5xx error code other than 401 is received.

6.4.9.3.2 GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 6.4.9.3.2-1 and 6.4.9.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.9.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

6.4.9.3.3 PUT

Not supported.

6.4.9.3.4 PATCH

Not supported.

6.4.9.3.5   DELETE

Not supported.

6.5   Data Model

6.5.1   Introduction

This clause defines the request and response data structures of the VNF Performance Management interface. If a request or response contains attributes not defined in the present specification, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

6.5.2   Resource and notification data types

6.5.2.1   Introduction

This clause defines the data structures to be used in resource representations and notifications.

6.5.2.2   Type: PmSubscriptionRequest

This type represents a subscription request. It shall comply with the provisions defined in table 6.5.2.2-1.

Table 6.5.2.2-1: Definition of the PmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	PmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

6.5.2.3   Type: PmSubscription

This type represents a subscription. It shall comply with the provisions defined in table 6.5.2.3-1.

Table 6.5.2.3-1: Definition of the PmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier that identifies the subscription
filter	PmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter
callbackUri	Uri	1	The URI of the endpoint to send the notification to
_links	Structure (inlined)	1	Links to resources related to this resource
>self	Link	1	URI of this resource

6.5.2.4   Type: ThresholdCrossedNotification

This type represents a notification that is sent when a threshold has been crossed. It shall comply with the provisions defined in table 6.5.2.4-1.

The notification shall be triggered by the VNFM when a threshold has been crossed.

Table 6.5.2.4-1: Definition of the ThresholdCrossedNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "ThresholdCrossedNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date and time of the generation of the notification.
thresholdId	Identifier	1	Identifier of the threshold which has been crossed.
crossingDirection	CrossingDirectionType	1	An indication of whether the threshold was crossed in upward or downward direction.
objectInstanceId	Identifier	1	Identifier of the measured object (i.e. a VNF instance).
subObjectInstanceId	IdentifierInVnf	0..1	Identifier of the sub-object of the measured object (i.e. a VNFC instance) to which the measurement applies. Shall be present if this is required in an external measurement specification. See note 1 and note 2.
performanceMetric	String	1	Performance metric associated with the threshold.
performanceValue	(any type)	1	Value of the metric that resulted in threshold crossing. See note 2.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	Link	1	Link to the related subscription.
>objectInstance	Link	0..1	Link to the resource representing the VNF instance to which the notified change applies. Shall be present if the VNF instance information is accessible as a resource.
>threshold	Link	1	Link to the resource that represents the threshold that was crossed.
NOTE 1: The sub-object allows to structure the measured object, but is not to be confused with sub-counters which allow to structure the measurement.			
NOTE 2: The type of the "performanceValue" attribute (i.e. scalar, structure (Object in JSON), or array (of scalars, arrays or structures/Objects)), and whether the measurement applies to sub-objects, is assumed to be defined in an external measurement specification.			

6.5.2.5 Type: PerformanceInformationAvailableNotification

This notification informs the receiver that performance information is available. It shall comply with the provisions defined in table 6.5.2.5-1.

The notification shall be triggered by the VNFM when new performance information collected by a PM job is available.

NOTE: The timing of sending this notification is determined by the capability of the producing entity to evaluate the threshold crossing condition.

Table 6.5.2.5-1: Definition of the PerformanceInformationAvailableNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "PerformanceInformationAvailableNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date and time of the generation of the notification.
objectInstanceId	Identifier	1	Identifier of the measured object (i.e. a VNF instance).
subObjectInstanceIds	IdentifierInVnf	0..N	Identifiers of the sub-objects of the measured object (i.e. VNFC instances) for which the measurements have been taken. Shall be present if the related PM job has been set up to measure only a subset of all VNFCs of the VNF instance, and shall be absent otherwise.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	Link	1	Link to the related subscription.
>objectInstance	Link	0..1	Link to the resource representing the VNF instance to which the notified change applies. Shall be present if the VNF instance information is accessible as a resource.
>pmJob	Link	1	Link to the resource that represents the PM job for which performance information is available.
>performanceReport	Link	1	Link from which the available performance information of data type "PerformanceReport" (see clause 6.5.2.10) can be obtained. This link should point to an "Individual performance report" resource as defined in clause 6.4.4.

6.5.2.6 Type: CreatePmJobRequest

This type represents a request to create a PM job. It shall comply with the provisions defined in table 6.5.2.6-1.

Table 6.5.2.6-1: Definition of the CreatePmJobRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceIds	Identifier	1..N	Identifiers of the measured objects (i.e. VNF instances) for which performance information is requested to be collected.
subObjectInstanceIds	IdentifierInVnf	0..N	Identifiers of the sub-objects of the measured object (i.e. VNFC instances) for which performance information is requested to be collected. May be present if the measurement can be taken per VNFC as defined in an external measurement specification. If this attribute is present, the cardinality of the "objectInstanceIds" attribute shall be 1.
criteria	PmJobCriteria	1	Criteria of the collection of performance information.

6.5.2.7 Type: PmJob

This type represents a PM job. It shall comply with the provisions defined in table 6.5.2.7-1.

Table 6.5.2.7-1: Definition of the PmJob data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this PM job.
objectInstanceIds	Identifier	1..N	Identifiers of the measured objects (i.e. VNF instances) for which performance information is collected.
subObjectInstanceIds	IdentifierInVnf	0..N	Identifiers of the sub-objects of the measured object (i.e. VNFC instances) for which performance information is requested to be collected. May be present if the measurement can be taken per VNFC as defined in an measurement specification. If this attribute is present, the cardinality of the "objectInstanceIds" attribute shall be 1.
criteria	PmJobCriteria	1	Criteria of the collection of performance information.
reports	Structure (inlined)	0..N	Information about available reports collected by this PM job.
>href	Uri	1	The Uri where the report can be obtained.
>readyTime	DateTime	1	The time when the report was made available.
>expiryTime	DateTime	0..1	The time when the report will expire.
>fileSize	UnsigendInt	0..1	The size of the report file in bytes, if known.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>objects	Link	0..N	Links to resources representing the VNF instances for which performance information is collected. Shall be present if the VNF instance information is accessible as a resource.

6.5.2.8   Type: CreateThresholdRequest

This type represents a request to create a threshold. It shall comply with the provisions defined in table 6.5.2.8-1.

Table 6.5.2.8-1: Definition of the CreateThresholdRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceId	Identifier	1	Identifier of the measured object (i.e. VNF instance) associated with this threshold.
subObjectInstanceIds	IdentifierInVnf	0..N	Identifiers of the sub-objects of the measured object (i.e. VNFC instances) associated with this threshold. May be present if the measurement can be taken per VNFC instance as defined in an external measurement specification.
criteria	ThresholdCriteria	1	Criteria that define this threshold.

6.5.2.9   Type: Threshold

This type represents a threshold. It shall comply with the provisions defined in table 6.5.2.9-1.

Table 6.5.2.9-1: Definition of the Threshold data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this threshold resource.
objectInstanceId	Identifier	1	Identifier of the measured object (i.e. VNF instance) associated with the threshold.
subObjectInstanceIds	IdentifierInVnf	0..N	Identifiers of the sub-objects of the measured object (i.e. VNFC instances) associated with the threshold. May be present if the measurement can be taken per VNFC instance as defined in an external measurement specification.
criteria	ThresholdCriteria	1	Criteria that define this threshold.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>object	Link	0..1	Link to a resource representing the VNF instance for which performance information is collected. Shall be present if the VNF instance information is accessible as a resource.

6.5.2.10   Type: PerformanceReport

This type defines the format of a performance report provided by the VNFM to the EM as a result of collecting performance information as part of a PM job. The type shall comply with the provisions defined in table 6.5.2.10-1.

Table 6.5.2.10-1: Definition of the PerformanceReport data type

Attribute name	Data type	Cardinality	Description
entries	Structure (inlined)	1..N	List of performance information entries. Each performance report entry is for a given metric of a given object (i.e. VNF instance), but can include multiple collected values.
>objectType	String	1	Defines the object type for which performance information is reported (i.e. VNF type). The string value shall be set to the vnfdId of the VNF instance to which the performance information relates.
>objectInstanceId	Identifier	1	Identifier of the measured object (i.e. VNF instance) for which the performance metric is reported.
>subObjectInstanceId	IdentifierInVnf	0..1	Identifier of the sub-object of the measured object (i.e. a VNFC instance) for which the performance metric is reported. Shall be present if this is required in the measurement specification. See notes 1 and note 2.
>performanceMetric	String	1	Name of the metric collected.
>performanceValue	Structure (inlined)	1..N	List of performance values with associated timestamp.
>>timeStamp	DateTime	1	Timestamp indicating when the data was collected.
>>value	(any type)	1	Value of the metric collected. See note 2.
NOTE 1: The sub-object allows to structure the measured object, but is not to be confused with sub-counters which allow to structure the measurement. EXAMPLE: Measured object: VnfInstanceXYZ Sub-object: VnfcInstance1 Measurement: vCPU_utilization Sub-counters: vCPU utilization of each of the vCPUs of VnfcInstance1 (vCPU_utilization.vCPU1, vCPU_utilization.vCPU2, etc.). NOTE 2: The type of the "value" attribute (i.e. scalar, structure (Object in JSON), or array (of scalars, arrays or structures/Objects)), and whether the measurement applies to sub-objects, is assumed to be defined in an external measurement specification outside the scope of the present document.			

6.5.3   Referenced structured data types

6.5.3.1   Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

6.5.3.2   Type: PmNotificationsFilter

This type represents a filter that can be used to subscribe for notifications related to performance management events. It shall comply with the provisions defined in table 6.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 6.5.3.2-1: Definition of the PmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: • ThresholdCrossedNotification • PerformanceInformationAvailableNotification See note.
NOTE: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.			

6.5.3.3   Type: PmJobCriteria

This type represents collection criteria for PM jobs. It shall comply with the provisions defined in table 6.5.3.3-1.

Table 6.5.3.3-1: Definition of the PmJobCriteria data type

Attribute name	Data type	Cardinality	Description
performanceMetric	String	0..N	This defines the types of performance metrics for the specified object instances, as specified in an external measurement specification. At least one of the two attributes (performance metric or group) shall be present.
performanceMetricGroup	String	0..N	Group of performance metrics. A metric group is a pre-defined list of metrics, known to the producer that it can decompose to individual metrics. At least one of the two attributes (performance metric or group) shall be present.
collectionPeriod	UnsignedInt	1	Specifies the periodicity at which the producer will collect performance information. The unit shall be seconds. See note 1 and note 2.
reportingPeriod	UnsignedInt	1	Specifies the periodicity at which the producer will report to the consumer. about performance information. The unit shall be seconds. See note 1 and note 2.
reportingBoundary	DateTime	0..1	Identifies a time boundary after which the reporting will stop. The boundary shall allow a single reporting as well as periodic reporting up to the boundary.

Attribute name	Data type	Cardinality	Description
NOTE 1:	At the end of each reportingPeriod, the producer will inform the consumer about availability of the performance data collected for each completed collection period during this reportingPeriod. The reportingPeriod should be equal to or a multiple of the collectionPeriod. In the latter case, the performance data for the collection periods within one reporting period are reported together.		
NOTE 2:	In particular when choosing short collection and reporting periods, the number of PM jobs that can be supported depends on the capability of the producing entity.		

6.5.3.4 Type: ThresholdCriteria

This type represents criteria that define a threshold. It shall comply with the provisions defined in table 6.5.3.4-1.

Table 6.5.3.4-1: Definition of the ThresholdCriteria data type

Attribute name	Data type	Cardinality	Description
performanceMetric	String	1	Defines the performance metric associated with the threshold, as specified in an external measurement specification.
thresholdType	Enum (inlined)	1	Type of threshold. This attribute determines which other attributes are present in the data structure. Permitted values: - SIMPLE: Single-valued static threshold See note 1.
simpleThresholdDetails	Structure (inlined)	0..1	Details of a simple threshold. Shall be present if thresholdType="SIMPLE".
>thresholdValue	Number	1	The threshold value. Shall be represented as a floating point number.
>hysteresis	Number	1	The hysteresis of the threshold. Shall be represented as a non-negative floating point number. A notification with crossing direction "UP" will be generated if the measured value reaches or exceeds "thresholdValue" + "hysteresis". A notification with crossing direction "DOWN" will be generated if the measured value reaches or undercuts "thresholdValue" - "hysteresis". See note 2.
NOTE 1:	In the present document, simple thresholds are defined. The definition of additional threshold types is left for future specification.		
NOTE 2:	The hysteresis is defined to prevent storms of threshold crossing notifications. When processing a request to create a threshold, implementations should enforce a suitable minimum value for this attribute (e.g. override the value or reject the request).		

6.5.4 Referenced simple data types and enumerations

6.5.4.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

6.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

6.5.4.3 Enumeration: CrossingDirectionType

The enumeration CrossingDirectionType shall comply with the provisions defined in table 6.5.4.3-1.

Table 6.5.4.3-1: Enumeration CrossingDirectionType

Enumeration value	Description
UP	The threshold was crossed in upward direction.
DOWN	The threshold was crossed in downward direction.

7   VNF Fault Management interface

7.1   Description

This interface allows the EM/VNF to subscribe to notifications regarding VNF alarms provided by the VNFM.

Virtualised resource alarms collected by the VNFM is filtered, correlated and modified by the VNFM and mapped to the corresponding VNF instance, resulting in alarms on that VNF instance which contain information on the VNFC(s) affected by the fault.

The operations provided through this interface are:

- Get Alarm List

- Acknowledge Alarm

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

- Escalate perceived severity

7.2   Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnffm" shall be used to represent {apiName}. The {apiVersion} shall be set to "v1" for the present document. All resource URIs in the clauses below are defined relative to the above base URI.

Figure 7.2-1 shows the overall resource URI structure defined for the VNF fault management interface.
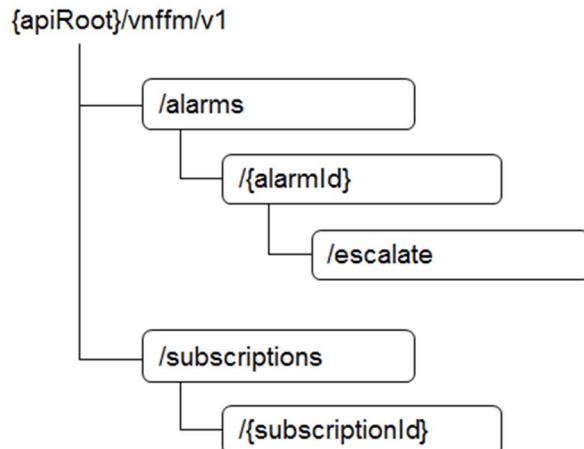
Figure 7.2-1: Resource URI structure of the VNF Fault Management interface

Table 7.2-1 lists the individual resources defined, and the applicable HTTP methods.

Table 7.2-1: Resources and methods overview of the VNF Fault Management interface
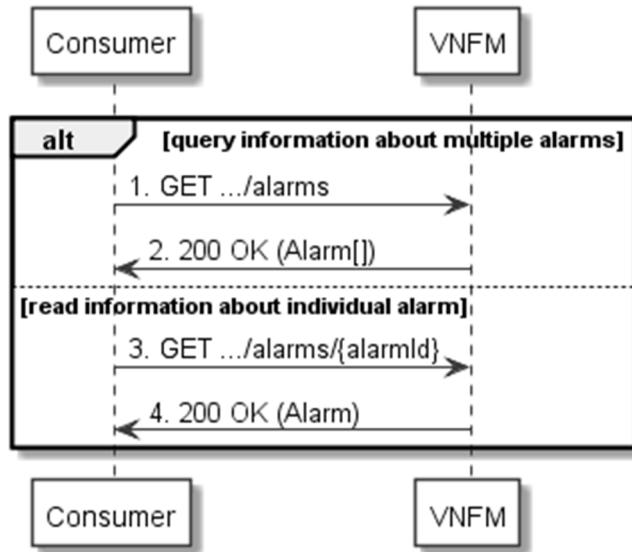
Resource name	Resource URI	HTTP Method	Meaning
Alarms	/alarms	GET	Query alarms related to VNF instances
Individual alarm	/alarms/{alarmId}	GET	Read individual alarm
		PATCH	Acknowledge individual alarm
Escalate perceived severity task	/alarms/{alarmId}/escalate	POST	Escalate the consumer's view of perceived severity
Subscriptions	/subscriptions	POST	Subscribe to VNF alarms
		GET	Query multiple subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	Read an individual subscription
		DELETE	Terminate a subscription
Notification endpoint	(client-provided)	POST	Notify about VNF alarms
		GET	Test the notification endpoint

7.3 Sequence diagrams (informative)

7.3.1 Flow of the Get Alarm List operation

This clause describes a sequence flow for querying one or multiple alarms.

NOTE: The consumer can be either EM or VNF depending on the operations.

Figure 7.3.1-1: Flow of alarm query/read

Alarm query, as illustrated in figure 7.3.1-1, consists of the following steps:

1) If the Consumer intends to query all alarms, it sends a GET request to the "Alarms" resource.

2) The VNFM returns a "200 OK" response to the Consumer, and includes zero or more data structures of type "Alarm" in the payload body.

3) If the Consumer intends to read a particular alarm, it sends a GET request to the "Individual alarm" resource, addressed by the appropriate alarm identifier in its resource URI.

4) The VNFM returns a "200 OK" response to the Consumer, and includes a data structure of type "Alarm" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

7.3.2   Escalate perceived severity task flow

This clause describes the procedure to escalate the consumer's view of the perceived severity for an alarm to the VNFM.

NOTE 1: The consumer can be either the EM or VNF.
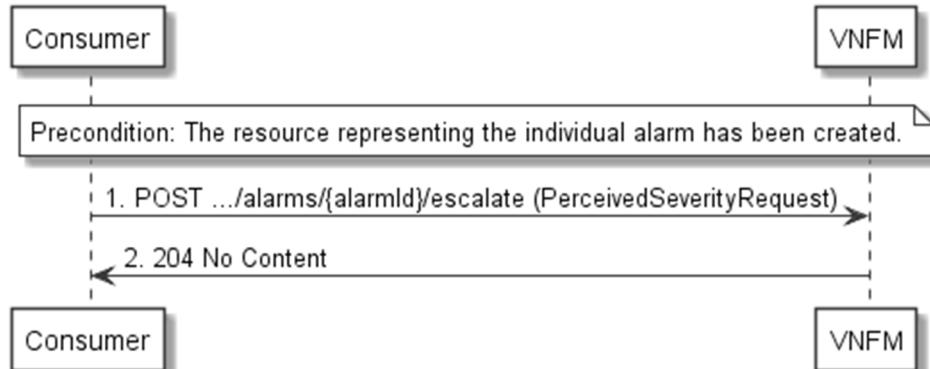
Figure 7.3.2-1: Escalate perceived severity task flow

Precondition: The resource representing the individual alarm has been created.

The escalate perceived severity task flow, as illustrated in figure 7.3.2-1, consists of the following steps:

1)   The Consumer sends a POST request to the "Escalate Perceived Severity" task resource of an individual alarm, which is identified by the "alarmId" in the resource URI.

2)   The VNFM returns a "204 No Content" response to the Consumer.

NOTE 2: If the value of the perceived severity is changed after this operation is finished, the VNFM will send an AlarmNotification to each subscribed Consumer.

Error handling: In case of failure, appropriate error information is provided in the response.

7.3.3   Flow of acknowledging alarm

This clause describes the procedure to acknowledge an individual alarm.
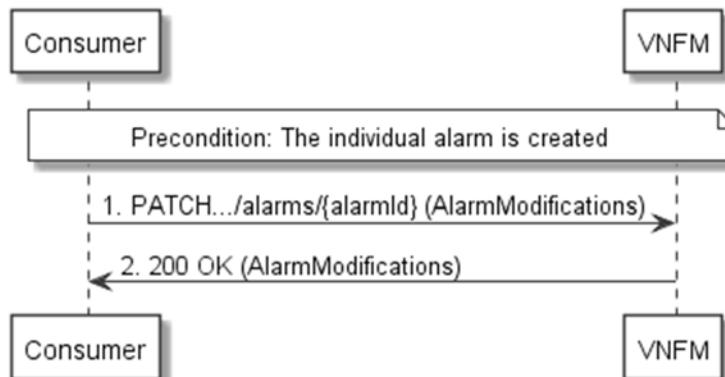
Figure 7.3.3-1: Flow of acknowledging alarm

Precondition: The resource representing the individual alarm has been created.

Acknowledge alarm, as illustrated in figure 7.3.3-1, consists of the following steps:

1)   The Consumer sends a PATCH request to the individual alarm.

2)   The VNFM returns a "200 OK" response to the Consumer, and includes a data structure of type "AlarmModifications" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

7.3.4   Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF fault management.
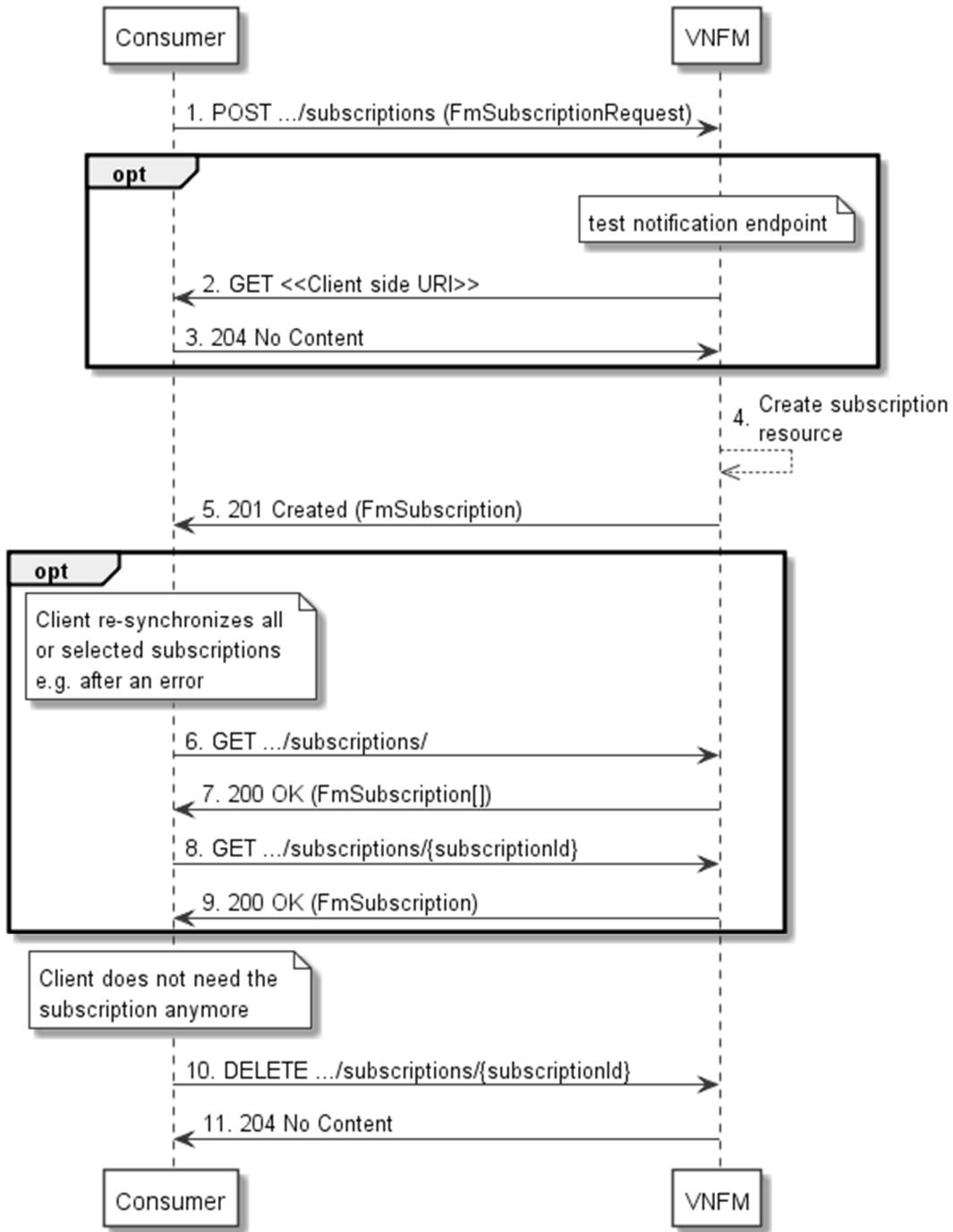
Figure 7.3.4-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 7.3.4-1:

1)   The Consumer sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "FmSubscriptionRequest". This data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2)   Optionally, to test the notification endpoint that was registered by the Consumer as part of the subscription, the VNFM sends a GET request to the notification endpoint URI.

3)   In that case, the Consumer returns a "204 No Content" response to indicate success.

4)   The VNFM creates a new subscription for notifications related to VNF fault management, and a resource that represents this subscription.

5)   The VNFM returns a "201 Created" response containing a data structure of type "FmSubscription", representing the subscription resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header.

6)   Optionally, for example when trying to recover from an error situation, the Consumer may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7)   In that case, the VNFM returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the Consumer.

8)   Optionally, for example when trying to recover from an error situation, the Consumer may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9)   In that case, the VNFM returns a "200 OK" response that contains a representation of that individual subscription.

10)   When the Consumer does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription.

11)   The VNFM acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The VNFM rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

7.3.5   Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF fault management.
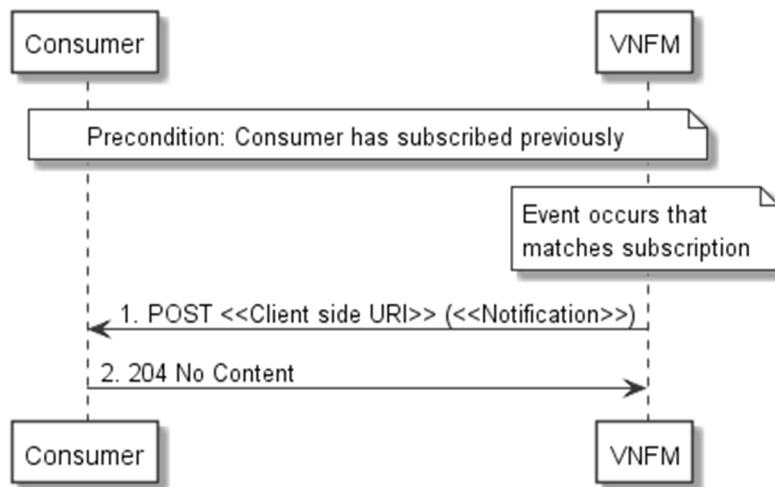
Figure 7.3.5-1: Flow of sending notifications

Precondition: The Consumer has subscribed previously for notifications related to VNF fault management.

The procedure consists of the following steps as illustrated in figure 7.3.5-1:

1)   If an event occurs that matches the filtering criteria which are part of the subscription, the VNFM generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the Consumer has registered as part of the subscription request. The variable <<Notification>> in the flow is a placeholder for the different types of notifications that can be sent by this API (see clauses 7.5.2.5, 7.5.2.6 and 7.5.2.7).

2)   The Consumer acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the Consumer, it can retry sending the notification.

7.4   Resources

7.4.1   Introduction

This clause defines all the resources and methods provided by the VNF fault management interface.

7.4.2   Resource: Alarms

7.4.2.1   Description

This resource represents a list of alarms related to VNF instances.

7.4.2.2   Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/alarms

This resource shall support the resource URI variables defined in table 7.4.2.2-1.

Table 7.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

7.4.2.3   Resource methods

7.4.2.3.1   POST

Not supported.

7.4.2.3.2   GET

The client can use this method to retrieve information about the alarm list.

This method shall follow the provisions specified in the tables 7.4.2.3.2-1 and 7.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. The VNF may supply its instance Id as an attribute filter. The following attribute names shall be supported in attribute-based filtering parameters: id, managedObjectId, vnfcInstanceIds, rootCauseFaultyResource.faultyResourceType, eventType, perceivedSeverity, probableCause. If the vnfcInstanceIds parameter is provided, exactly one value for the managedObjectId attribute shall be provided.

NOTE: There are no attribute selectors defined for this resource as the Alarm attributes with cardinality 0..1 or 0..N are not structurally complex in nature.

Table 7.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	Alarm	0..N	200 OK	The request has succeeded. The response body shall contain the list of related alarms.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.2.3.3   PUT

Not supported.

7.4.2.3.4   PATCH

Not supported.

7.4.2.3.5   DELETE

Not supported.

7.4.3   Resource: Individual alarm

7.4.3.1   Description

This resource represents an individual alarm.

7.4.3.2   Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/alarms/{alarmId}

This resource shall support the resource URI variables defined in table 7.4.3.2-1.

Table 7.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
alarmId	Identifier of the alarm. See note.
NOTE: This identifier can be retrieved from the "id" attribute of the "alarm" attribute in the AlarmNotification or AlarmClearedNotification. It can also be retrieved from the "id" attribute of the applicable array element in the payload body of the response to a GET request to the "Alarms" resource.	

7.4.3.3 Resource methods

7.4.3.3.1 POST

Not supported.

7.4.3.3.2 GET

The client can use this method to read an individual alarm.

This method shall follow the provisions specified in the tables 7.4.3.3.2-1 and 7.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	Alarm	1	200	Information about an individual alarm was read successfully. The response body shall contain a representation of the individual alarm.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.3.3.3 PUT

Not supported.

7.4.3.3.4 PATCH

This method modifies an individual alarm resource.

This method shall follow the provisions specified in the tables 7.4.3.3.4-1 and 7.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.3.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality		Description
Request body	AlarmModifications	1		The parameter for the alarm modification, as defined in clause 7.5.2.9. The Content-Type header shall be set to "application/merge-patch+json" according to IETF RFC 7396 [15].

	Data type	Cardinality	Response Codes	Description
Response body	AlarmModifications	1	200 OK	The request was accepted and completed. The response body shall contain attribute modifications for an "Individual alarm" resource (see clause 7.5.2.4).
	ProblemDetails	0..1	409 Conflict	Error: The operation cannot be executed currently, due to a conflict with the state of the "Individual alarm" resource. Typically, this is due to the fact that the alarm is already in the state that is requested to be set (such as trying to acknowledge an already-acknowledged alarm). The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	412 Precondition failed	Error: A precondition given in an HTTP request header is not fulfilled. Typically, this is due to an ETag mismatch, indicating that the resource was modified by another entity. The response body should contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.3.3.5   DELETE

Not supported.

7.4.4   Resource: Escalate Perceived Severity task

7.4.4.1   Description

This task resource represents the "Escalate Perceived Severity" operation. The client can use this resource to escalate the perceived severity of an alarm with the VNFM. This operation does not directly modify the value of perceived severity attribute in the alarm information element within the VNFM. VNFM implementation (e.g. controlled by operator configuration) will determine how it should act upon receipt of the requested change in perceived severity. Some requests from the EM/VNF may be respected and applied directly by the VNFM, while others may be ignored by the VNFM.

7.4.4.2   Resource definition

The resource URI is:

{apiRoot}/vnflcm/v1/alarms/{alarmId}/escalate

This resource shall support the resource URI variables defined in table 7.4.4.2-1.

Table 7.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
alarmId	Identifier of the alarm.

7.4.4.3   Resource Methods

7.4.4.3.1   POST

The POST method enables the consumer to escalate the perceived severity of an alarm that is represented by an individual alarm resource.

This method shall follow the provisions specified in the tables 7.4.4.3.1-1 and 7.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.4.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	PerceivedSeverityRequest	1	The proposed "escalated perceived severity" value, as defined in clause 7.5.2.7.	
Response body	Data type	Cardinality	Response Codes	Description
	n/a	1	204 No Content	The VNFM has received the proposed "escalated perceived severity" value successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	Any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.4.3.2   GET

Not supported.

7.4.4.3.3   PUT

Not supported.

7.4.4.3.4   PATCH

Not supported.

7.4.4.3.5   DELETE

Not supported.

7.4.5   Resource: Subscriptions

7.4.5.1   Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF alarms and to query its subscriptions.

7.4.5.2   Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/subscriptions

This resource shall support the resource URI variables defined in table 7.4.5.2-1.

Table 7.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

7.4.5.3   Resource methods

7.4.5.3.1   POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 7.4.5.3.1-1 and 7.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the EM or VNF, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 7.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
none supported		

Table 7.4.5.3.1-2: Details of the POST request/response on this resource

<table>
<tr><td rowspan="2">Request body</td><td>Data type</td><td>Cardinality</td><td colspan="2">Description</td></tr>
<tr><td>FmSubscriptionRequest</td><td>1</td><td colspan="2">Details of the subscription to be created, as defined in clause 7.5.2.2.</td></tr>
<tr><td rowspan="7">Response body</td><td>Data type</td><td>Cardinality</td><td>Response Codes</td><td>Description</td></tr>
<tr><td>FmSubscription</td><td>1</td><td>201 Created</td><td>The subscription was created successfully.
The response body shall contain a representation of the created subscription resource.
The HTTP response shall include a "Location:" HTTP header that points to the created subscription resource.</td></tr>
<tr><td>n/a</td><td></td><td>303 See Other</td><td>A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions.
The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource.
The response body shall be empty.</td></tr>
<tr><td>ProblemDetails</td><td>See clauses 4.3.5.4/ 4.3.5.5</td><td>4xx/5xx</td><td>In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.</td></tr>
</table>

7.4.5.3.2   GET

The client can use this method to retrieve the list of active subscriptions for VNF alarms subscribed by the client. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 7.4.5.3.2-1 and 7.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The VNFM shall support receiving filtering parameters as part of the URI query string. The EM may supply filtering parameters. The VNF may supply its instance Id as an attribute filter. All attribute names that appear in the FmSubscription and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 7.4.5.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	FmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain the representations of all active subscriptions of the functional block that invokes the method.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.5.3.3   PUT

Not supported.

7.4.5.3.4   PATCH

Not supported.

7.4.5.3.5   DELETE

Not supported.

7.4.6   Resource: Individual subscription

7.4.6.1   Description

This resource represents an individual subscription for VNF alarms. The client can use this resource to read and to terminate a subscription to notifications related to VNF fault management.

7.4.6.2   Resource definition

The resource URI is:

{apiRoot}/vnffm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 7.4.6.2-1.

Table 7.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE:	This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.

7.4.6.3   Resource methods

7.4.6.3.1   POST

Not supported.

7.4.6.3.2   GET

The client can use this method for reading an individual subscription for VNF alarms subscribed by the client.

This method shall follow the provisions specified in the tables 7.4.6.3.2-1 and 7.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.6.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	FmSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

7.4.6.3.3   PUT

Not supported.

7.4.6.3.4   PATCH

Not supported.

7.4.6.3.5   DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the tables 7.4.6.3.5-1 and 7.4.6.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.6.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.7 Resource: Notification endpoint

7.4.7.1 Description

This resource represents a notification endpoint for VNF alarms. The API producer can use this resource to send notifications related to VNF alarms or about a rebuilt alarm list to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

7.4.7.2 Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 7.4.7.2-1.

Table 7.4.7.2-1: Resource URI variables for this resource

Name	Definition
n/a	

7.4.7.3 Resource methods

7.4.7.3.1 POST

The POST method notifies a VNF alarm or that the alarm list has been rebuilt.

This method shall follow the provisions specified in the tables 7.4.7.3.1-1 and 7.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one of the alternatives defined in table 7.4.7.3.1-2.

Table 7.4.7.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description
Request body	AlarmNotification	1	Information of a VNF alarm.
	AlarmClearedNotification	1	Information of the clearance of a VNF alarm.
	AlarmListRebuiltNotification	1	Information that the alarm list has been rebuilt by the VNFM.
	Data type	Cardinality	Response Codes	Description
Response body	n/a		204 No Content	The notification was delivered successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5 may be returned. The VNFM shall not retry sending the notification if a 4xx/5xx error code other than 401 is received.

7.4.7.3.2   GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 7.4.7.3.2-1 and 7.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.7.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description
Request body	n/a		
	Data type	Cardinality	Response Codes	Description
Response body	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

7.4.7.3.3   PUT

Not supported.

7.4.7.3.4   PATCH

Not supported.

7.4.7.3.5 DELETE

Not supported.

7.5 Data Model

7.5.1 Introduction

This clause defines the request and response data structures of the VNF fault management interface. If a request or response contains attributes not defined in the present specification, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

7.5.2 Resource and notification data types

7.5.2.1 Introduction

This clause defines the data structures to be used in the resource representations and notifications for the VNF fault management interface.

7.5.2.2 Type: FmSubscriptionRequest

This type represents a subscription request related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.2.2-1.

Table 7.5.2.2-1: Definition of the FmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	FmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

7.5.2.3 Type: FmSubscription

This type represents a subscription related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.2.3-1.

Table 7.5.2.3-1: Definition of the FmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	FmNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.

## 7.5.2.4		Type: Alarm

The alarm data type encapsulates information about an alarm. It shall comply with the provisions defined in table 7.5.2.4-1.

Table 7.5.2.4-1: Definition of the Alarm data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this Alarm information element.
managedObjectId	Identifier	1	Identifier of the affected VNF instance.
vnfcInstanceIds	IdentifierInVnf	1..N	Identifiers of the affected VNFC instances.
rootCauseFaultyResource	FaultyResourceInfo	1	The virtualised resources that are causing the VNF fault.
alarmRaisedTime	DateTime	1	Time stamp indicating when the alarm is raised by the managed object.
alarmChangedTime	DateTime	0..1	Time stamp indicating when the alarm was last changed. It shall be present if the alarm has been updated.
alarmClearedTime	DateTime	0..1	Time stamp indicating when the alarm was cleared. It shall be present if the alarm has been cleared.
ackState	Enum (inlined)	1	Acknowledgement state of the alarm. Permitted values: UNACKNOWLEDGED ACKNOWLEDGED.
perceivedSeverity	PerceivedSeverityType	1	Perceived severity of the managed object failure.
eventTime	DateTime	1	Time stamp indicating when the fault was observed.
eventType	EventType	1	Type of event.
faultType	String	0..1	Additional information to clarify the type of the fault.
probableCause	String	1	Information about the probable cause of the fault.
isRootCause	Boolean	1	Attribute indicating if this fault is the root for other correlated alarms. If TRUE, then the alarms listed in the attribute CorrelatedAlarmId are caused by this fault.
correlatedAlarmIds	Identifier	0..N	List of identifiers of other alarms correlated to this fault.
faultDetails	String	0..N	Provides additional information about the fault.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>objectInstance	Link	0..1	Link to the resource representing the VNF instance to which the notified alarm is correlated. Shall be present if the VNF instance information is accessible as a resource.

## 7.5.2.5		Type: AlarmNotification

This type represents an alarm notification about VNF faults. It shall comply with the provisions defined in table 7.5.2.5-1.

The notification shall be triggered by the VNFM when:

- An alarm has been created

- An alarm has been updated, e.g. if the severity of the alarm has changed

Table 7.5.2.5-1: Definition of the AlarmNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
alarm	Alarm	1	Information about an alarm including AlarmId, affected VNF identifier, and FaultDetails.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	Link	1	Link to the related subscription.

7.5.2.6   Type: AlarmClearedNotification

This type represents an alarm cleared notification about VNF faults. It shall comply with the provisions defined in table 7.5.2.6-1.

The notification shall be triggered by the VNFM when an alarm has been cleared.

Table 7.5.2.6-1: Definition of the AlarmClearedNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmClearedNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
alarmId	Identifier	1	Alarm identifier.
alarmClearedTime	DateTime	1	The time stamp indicating when the alarm was cleared.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	Link	1	Link to the related subscription.
>alarm	Link	1	Link to the resource that represents the related alarm.

7.5.2.7   Type: PerceivedSeverityRequest

This type represents the escalated value of the perceived severity for an alarm. It shall comply with the provisions defined in table 7.5.2.7-1.

Table 7.5.2.7-1: Definition of the PerceivedSeverityRequest data type

Attribute name	Data type	Cardinality	Description
proposedPerceived Severity	PerceivedSeverityType	1	Indicates the proposed escalated perceived severity for an alarm.

7.5.2.8   Type: AlarmListRebuiltNotification

This type represents a notification that the alarm list has been rebuilt, e.g. if the VNFM detects its storage holding the alarm list is corrupted. It shall comply with the provisions defined in table 7.5.2.8-1.

The notification shall be triggered by the VNFM when the alarm list has been rebuilt.

Table 7.5.2.8-1: Definition of the AlarmListRebuiltNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "AlarmListRebuiltNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	Link	1	Link to the related subscription.
>alarms	Link	1	Link to the alarm list, i.e. the Alarms resource.

7.5.2.9   Type: AlarmModifications

This type represents attribute modifications for an "Individual alarm" resource, i.e. modifications to a resource representation based on the "Alarm" data type. The attributes of "Alarm" that can be modified according to the provisions in clause 7.5.2.4 are included in the "AlarmModifications" data type.

The "AlarmModifications" data type shall comply with the provisions defined in table 7.5.2.9-1.

Table 7.5.2.9-1: Definition of the AlarmModifications data type

Attribute name	Data type	Cardinality	Description
ackState	Enum (inlined)	1	New value of the "ackState" attribute in "Alarm". Permitted values: - ACKNOWLEDGED.

7.5.3   Referenced structured data types

7.5.3.1   Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

7.5.3.2   Type: FmNotificationsFilter

This type represents a subscription filter related to notifications about VNF faults. It shall comply with the provisions defined in table 7.5.3.2-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 7.5.3.2-1: Definition of the FmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: • AlarmNotification • AlarmClearedNotification • AlarmListRebuiltNotification See note.
faultyResourceTypes	FaultyResourceType	0..N	Match VNF alarms with a faulty resource type listed in this attribute.
perceivedSeverities	PerceivedSeverityType	0..N	Match VNF alarms with a perceived severity listed in this attribute.
eventTypes	EventType	0..N	Match VNF alarms with an event type listed in this attribute.
probableCauses	String	0..N	Match VNF alarms with a probable cause listed in this attribute.
NOTE:	The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems.		

7.5.3.3   Type: FaultyResourceInfo

This type represents the faulty virtual resources that have a negative impact on a VNF. It shall comply with the provisions defined in table 7.5.3.3-1.

Table 7.5.3.3-1: Definition of the FaultyResourceInfo data type

Attribute name	Data type	Cardinality	Description
faultyResource	ResourceHandle	1	Information that identifies the faulty resource instance and its managing entity.
faultyResourceType	FaultyResourceType	1	Type of the faulty resource.

7.5.4   Referenced simple data types and enumerations

7.5.4.1   Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in the previous clauses.

7.5.4.2   Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.

7.5.4.3   Enumeration: PerceivedSeverityType

The enumeration PerceivedSeverityType shall comply with the provisions defined in table 7.5.4.3-1. It indicates the relative level of urgency for operator attention.

Table 7.5.4.3-1: Enumeration PerceivedSeverityType

Enumeration value	Description
CRITICAL	The Critical severity level indicates that a service affecting condition has occurred and an immediate corrective action is required. Such a severity can be reported, for example, when a managed object becomes totally out of service and its capability needs to be restored (Recommendation ITU-T X.733 [18]).
MAJOR	The Major severity level indicates that a service affecting condition has developed and an urgent corrective action is required. Such a severity can be reported, for example, when there is a severe degradation in the capability of the managed object and its full capability needs to be restored (Recommendation ITU-T X.733 [18]).
MINOR	The Minor severity level indicates the existence of a non-service affecting fault condition and that corrective action should be taken in order to prevent a more serious (for example, service affecting) fault. Such a severity can be reported, for example, when the detected alarm condition is not currently degrading the capacity of the managed object (Recommendation ITU-T X.733 [18]).
WARNING	The Warning severity level indicates the detection of a potential or impending service affecting fault, before any significant effects have been felt. Action should be taken to further diagnose (if necessary) and correct the problem in order to prevent it from becoming a more serious service affecting fault (Recommendation ITU-T X.733 [18]).
INDETERMINATE	The Indeterminate severity level indicates that the severity level cannot be determined (Recommendation ITU-T X.733 [18]).
CLEARED	The Cleared severity level indicates the clearing of one or more previously reported alarms. This alarm clears all alarms for this managed object that have the same Alarm type, Probable cause and Specific problems (if given) (Recommendation ITU-T X.733 [18]).

7.5.4.4   Enumeration: EventType

The enumeration EventType represents those types of events that trigger an alarm. It shall comply with the provisions defined in table 7.5.4.4-1.

Table 7.5.4.4-1: Enumeration EventType

Enumeration value	Description
COMMUNICATIONS_ALARM	An alarm of this type is associated with the procedure and/or process required conveying information from one point to another (Recommendation ITU-T X.733 [18]).
PROCESSING_ERROR_ALARM	An alarm of this type is associated with a software or processing fault (Recommendation ITU-T X.733 [18]).
ENVIRONMENTAL_ALARM	An alarm of this type is associated with a condition related to an enclosure in which the equipment resides (Recommendation ITU-T X.733 [18]).
QOS_ALARM	An alarm of this type is associated with degradation in the quality of a service (Recommendation ITU-T X.733 [18]).
EQUIPMENT_ALARM	An alarm of this type is associated with an equipment fault (Recommendation ITU-T X.733 [18]).

7.5.4.5   Enumeration: FaultyResourceType

The enumeration FaultyResourceType represents those types of faulty resource. It shall comply with the provisions defined in table 7.5.4.5-1.

Table 7.5.4.5-1: Enumeration FaultyResourceType

Enumeration value	Description
COMPUTE	Virtual compute resource
STORAGE	Virtual storage resource
NETWORK	Virtual network resource

8   VNF Indicator interface

8.1   Description

This interface allows the EM/VNF to provide information on value changes of VNF related indicators. VNF related indicators are declared in the VNFD.

The operations provided through this interface are:

- Get Indicator Value

- Subscribe

- Query Subscription Information

- Terminate Subscription

- Notify

8.2   Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnfind" shall be used to represent {apiName}. The {apiVersion} shall be set to "v1" for the present document. All resource URIs in the clauses below are defined relative to the above base URI.

Figure 8.2-1 shows the overall resource URI structure defined for the VNF Indicator interface.
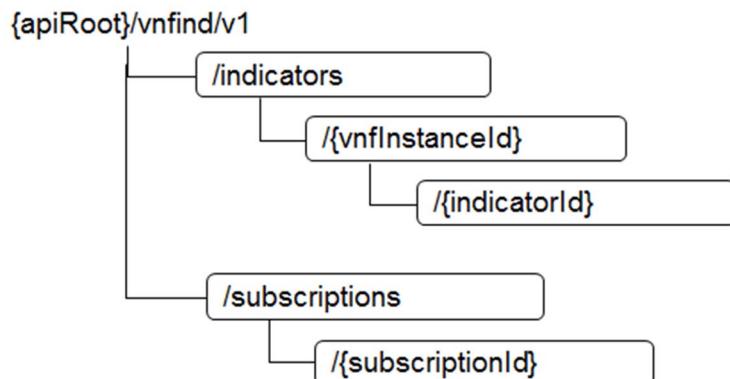
Figure 8.2-1: Resource URI structure of the VNF Indicator Interface

Table 8.2-1 lists the individual resources defined, and the applicable HTTP methods.

Table 8.2-1: Resources and methods overview of the VNF Indicator interface
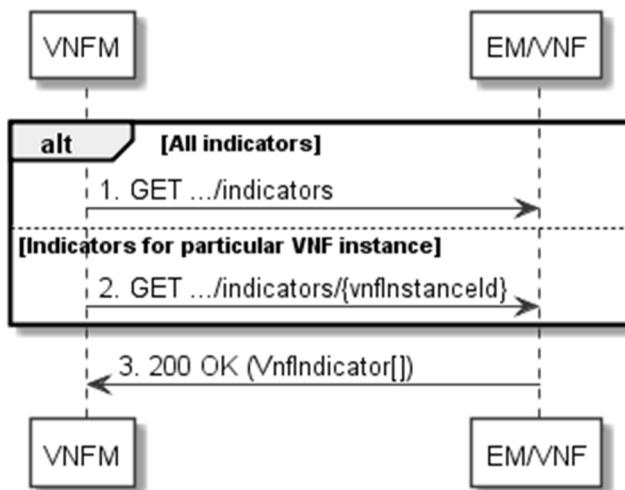
Resource name	Resource URI	HTTP Method	Meaning
VNF indicators	/indicators	GET	Query multiple VNF indicators. See note
VNF indicators related to a VNF instance	/indicators/{vnfInstanceId}	GET	Query multiple VNF indicators related to one VNF instance
Individual VNF indicator	/indicators/{vnfInstanceId}/{indicatorId}	GET	Read an individual VNF indicator
Subscriptions	/subscriptions	POST	Subscribe to VNF indicator change notifications
		GET	Query multiple subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	Read an individual subscription
		DELETE	Terminate a subscription
Notification endpoint	(client-provided)	POST	Notify about VNF indicator change
		GET	Test the notification endpoint
NOTE: This resource allows to query all VNF indicators that are known to the VNFM.			

8.3   Sequence diagrams (informative)

8.3.1   Flow of querying VNF indicators

This clause describes a sequence for querying VNF indicators.

Figure 8.3.1-1: Flow of querying VNF indicators

VNF indicator query, as illustrated in figure 8.3.1-1, consists of the following steps:

1)   If the VNFM intends to query all VNF indicators, it sends a GET request to the "VNF indicators" resource.

2)   If the VNFM intends to query the VNF indicators of a particular VNF instance, it sends a GET request to the "VNF indicators related to a VNF instance" resource.

3)   The EM/VNF returns a "200 OK" response to the VNFM, and includes zero or more data structures of type "VnfIndicator" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

8.3.2   Flow of reading a VNF indicator

This clause describes a sequence for reading a VNF indicator, i.e. for getting the indicator value.
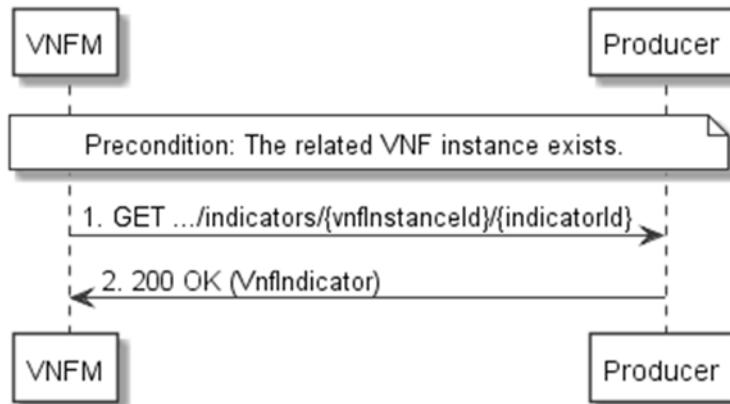
Figure 8.3.2-1: Flow of reading a VNF indicator

Precondition: The related VNF instance exists.

Reading a VNF indicator, as illustrated in figure 8.3.2-1, consists of the following steps:

1)   The VNFM sends a GET request to the "Individual VNF indicator" resource that is to be read.

2)   The EM/VNF returns a "200 OK" response to the VNFM, and includes a data structure of type "VnfIndicator" in the payload body.

Error handling: In case of failure, appropriate error information is provided in the response.

8.3.3   Flow of managing subscriptions

This clause describes the procedure for creating, querying/reading and terminating subscriptions to notifications related to VNF indicator value changes.
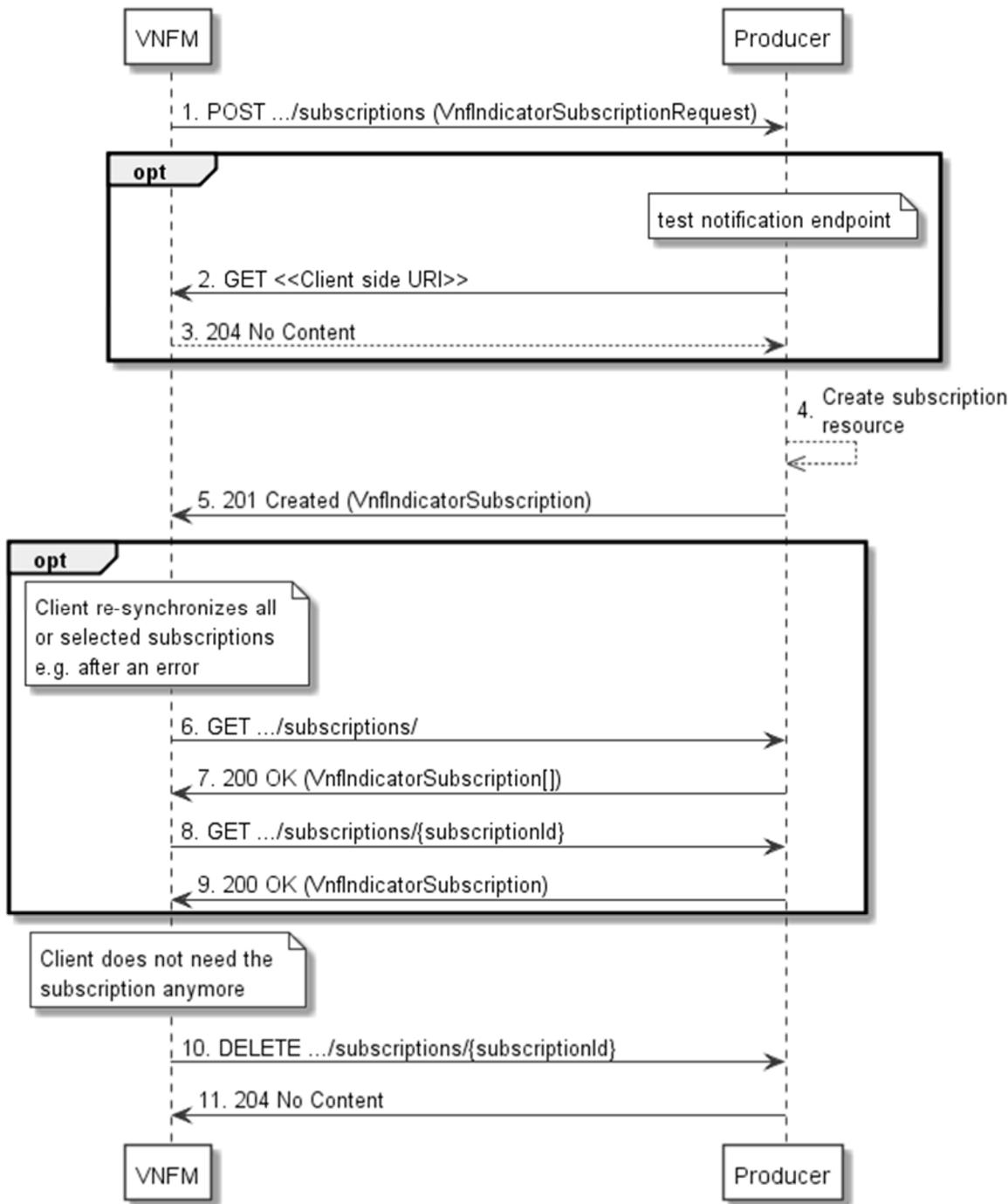
Figure 8.3.3-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in figure 8.3.3-1:

1) The VNFM sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "VnfIndicatorSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the VNFM will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the VNFM as part of the subscription, the Producer sends a GET request to the notification endpoint URI.

3) In that case, the Producer returns a "204 No Content" response to indicate success.

4) The Producer creates a new subscription to notifications related to VNF indicator value changes, and a resource that represents this subscription.

5) The Producer returns a 201 Created response containing a data structure of type "VnfIndicatorSubscription" representing the subscription resource just created by the Producer, and provides the URI of the newly-created resource in the "Location:" HTTP header.

6) If desired, e.g. to recover from an error situation, the VNFM may query information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the Producer returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the VNFM.

8) If desired, e.g. to recover from an error situation, the VNFM may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

9) In that case, the Producer returns a "200 OK" response that contains a representation of that individual subscription.

10) If the VNFM does not need the subscription anymore, it terminates the subscription by sending a DELETE request to the resource that represents the individual subscription to remove.

11) The VNFM acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The Producer rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

8.3.4   Flow of sending notifications

This clause describes the procedure for sending notifications.
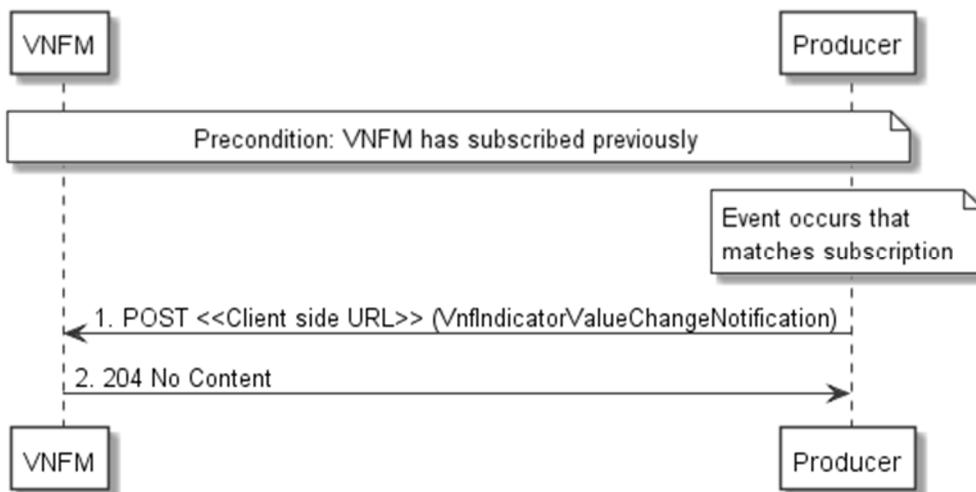
Figure 8.3.4-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in figure 8.3.4-1:

Precondition: The VNFM has subscribed previously to notifications related to VNF indicator value changes.

1) If an event occurs that matches the filtering criteria which are part of the subscription, the Producer generates an VnfIndicatorValueChangeNotification that includes information about the event, and sends it in the body of a POST request to the client side URI which the VNFM has registered as part of the subscription request.

2) The VNFM acknowledges the successful delivery of the notification by returning a "204 No Content" response.

Error handling: If the VNFM does not receive the "204 No Content" response from the Producer, it can retry sending the notification.

8.4   Resources

8.4.1   Introduction

This clause defines all the resources and methods provided by the VNF Indicator interface.

8.4.2   Resource: VNF indicators

8.4.2.1   Description

This resource represents VNF indicators. The client can use this resource to query multiple VNF indicators.

8.4.2.2   Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators

This resource shall support the resource URI variables defined in table 8.4.2.2-1.

Table 8.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

8.4.2.3   Resource methods

8.4.2.3.1   POST

Not supported.

8.4.2.3.2   GET

The GET method queries multiple VNF indicators.

This method shall follow the provisions specified in the tables 8.4.2.3.2-1 and 8.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The EM/VNF shall support receiving filtering parameters as part of the URI query string. The VNFM may supply filtering parameters. All attribute names that appear in the VnfIndicator data type and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 8.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	VnfIndicator	0..N	200 OK	The list of VNF indicators was queried successfully. The response body shall contain the representations of all VNF indicators that match the attribute-based filtering parameters.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.2.3.3   PUT

Not supported.

8.4.2.3.4   PATCH

Not supported.

8.4.2.3.5   DELETE

Not supported.

8.4.3   Resource: VNF indicators related to a VNF instance

8.4.3.1   Description

This resource represents VNF indicators related to a VNF instance. The client can use this resource to query multiple VNF indicators that are related to a particular VNF instance.

8.4.3.2   Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators/{vnfInstanceId}

This resource shall support the resource URI variables defined in table 8.4.3.2-1.

Table 8.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to which the VNF indicator applies. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

8.4.3.3   Resource methods

8.4.3.3.1   POST

Not supported.

8.4.3.3.2   GET

The GET method queries multiple VNF indicators related to a VNF instance.

This method shall follow the provisions specified in the tables 8.4.3.3.2-1 and 8.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The EM/VNF shall support receiving filtering parameters as part of the URI query string. The VNFM may supply filtering parameters. All attribute names that appear in the VnfIndicator data type and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 8.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**	
	VnfIndicator	0..N	200 OK	The list of VNF indicators was queried successfully. The response body shall contain the representations of all VNF indicators that are related to the particular VNF instance and that match the attribute filter.	
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

8.4.3.3.3   PUT

Not supported.

8.4.3.3.4   PATCH

Not supported.

8.4.3.3.5   DELETE

Not supported.

8.4.4   Resource: Individual VNF indicator

8.4.4.1   Description

This resource represents an individual VNF indicator. The client can use this resource to read an individual VNF indicator.

8.4.4.2   Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/indicators/{vnfInstanceId}/{indicatorId}

This resource shall support the resource URI variables defined in table 8.4.4.2-1.

Table 8.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfInstanceId	Identifier of the VNF instance to which the VNF indicator applies. See note 1.
indicatorId	Identifier of the VNF indicator. See note 2.
NOTE 1: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new VNF instance resource. It can also be retrieved from the "id" attribute in the payload body of that response. NOTE 2: This identifier can be retrieved from the resource referenced by the payload body in the response to a POST request creating a new VNF instance resource.	

8.4.4.3   Resource methods

8.4.4.3.1   POST

Not supported.

8.4.4.3.2   GET

The GET method reads a VNF indicator.

This method shall follow the provisions specified in the tables 8.4.4.3.2-1 and 8.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.4.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description	
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	VnfIndicator	1	200 OK	The VNF indicator was read successfully. The response body shall contain the representation of the VNF indicator.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

8.4.4.3.3   PUT

Not supported.

8.4.4.3.4   PATCH

Not supported.

8.4.4.3.5   DELETE

Not supported.

8.4.5   Resource: Subscriptions

8.4.5.1   Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to VNF indicator value changes, and to query its subscriptions.

8.4.5.2   Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/subscriptions

This resource shall support the resource URI variables defined in table 8.4.5.2-1.

Table 8.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

8.4.5.3   Resource methods

8.4.5.3.1   POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the tables 8.4.5.3.1-1 and 8.4.5.3.1-2 for URI query parameters, request and response data structures, and response codes.

Creation of two subscription resources with the same callbackURI and the same filter can result in performance degradation and will provide duplicates of notifications to the VNFM, and might make sense only in very rare use cases. Consequently, the VNFM may either allow creating a subscription resource if another subscription resource with the same filter and callbackUri already exists (in which case it shall return the "201 Created" response code), or may decide to not create a duplicate subscription resource (in which case it shall return a "303 See Other" response code referencing the existing subscription resource with the same filter and callbackUri).

Table 8.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.5.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	VnfIndicatorSubscription Request	1		Details of the subscription to be created.
	Data type	**Cardinality**	**Response Codes**	**Description**
Response body	VnfIndicatorSubscription	1	201 Created	The subscription was created successfully. The response body shall contain a representation of the created subscription resource. The HTTP response shall include a "Location" HTTP header that points to the created subscription resource.
	n/a		303 See Other	A subscription with the same callbackURI and the same filter already exists and the policy of the VNFM is to not create redundant subscriptions. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the existing subscription resource. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.5.3.2 GET

The GET method queries the list of active subscriptions of the functional block that invokes the method. It can be used e.g. for resynchronization after error situations.

This method shall follow the provisions specified in the tables 8.4.5.3.2-1 and 8.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
(filter)	0..1	Attribute-based filtering parameters according to clause 4.3.2. The EM/VNF shall support receiving filtering parameters as part of the URI query string. The VNFM may supply filtering parameters. All attribute names that appear in the VnfIndicatorSubscription data type and in data types referenced from it shall be supported in attribute-based filtering parameters.

Table 8.4.5.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Description
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfIndicatorSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain the representations of all active subscriptions of the functional block that invokes the method which match the attribute filter.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering parameters. The response body shall contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.5.3.3 PUT

Not supported.

8.4.5.3.4 PATCH

Not supported.

8.4.5.3.5 DELETE

Not supported.

8.4.6 Resource: Individual subscription

8.4.6.1 Description

This resource represents an individual subscription. The client can use this resource to read and to terminate a subscription to notifications related to VNF indicator value changes.

8.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnfind/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in table 8.4.6.2-1.

Table 8.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
subscriptionId	Identifier of this subscription. See note.
NOTE: This identifier can be retrieved from the resource referenced by the "Location" HTTP header in the response to a POST request creating a new subscription resource. It can also be retrieved from the "id" attribute in the payload body of that response.	

8.4.6.3   Resource methods

8.4.6.3.1   POST

Not supported.

8.4.6.3.2   GET

The GET method reads an individual subscription.

This method shall follow the provisions specified in the tables 8.4.6.3.2-1 and 8.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.6.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	VnfIndicatorSubscription	1	200 OK	The operation has completed successfully. The response body shall contain a representation of the subscription resource.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

8.4.6.3.3   PUT

Not supported.

8.4.6.3.4   PATCH

Not supported.

8.4.6.3.5   DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the tables 8.4.6.3.5-1 and 8.4.6.3.3-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.6.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Description	
	n/a			
Response body	Data type	Cardinality	Response Codes	Description
	n/a		204 No Content	The subscription resource was deleted successfully. The response body shall be empty.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

8.4.7   Resource: Notification endpoint

8.4.7.1   Description

This resource represents a notification endpoint. The API producer can use this resource to send notifications related to VNF indicator value changes to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

8.4.7.2   Resource definition

The resource URI is provided by the client when creating the subscription.

This resource shall support the resource URI variables defined in table 8.4.7.2-1.

Table 8.4.7.2-1: Resource URI variables for this resource

Name	Definition
none supported	

8.4.7.3   Resource methods

8.4.7.3.1   POST

The POST method delivers a notification from the server to the client.

This method shall follow the provisions specified in the tables 8.4.7.3.1-1 and 8.4.7.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Each notification request body shall include exactly one instance of the VnfIndicatorValueChangeNotification structure.

Table 8.4.7.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description		
	VnfIndicatorValueChangeNotification	1	A notification about VNF indicator value changes.		
Response body	Data type	Cardinality	Response Codes	Description	
	n/a		204 No Content	The notification was delivered successfully. The response body shall be empty.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned. The API producer shall not retry sending the notification if a 4xx/5xx error code other than 401 is received.	

8.4.7.3.2   GET

The GET method allows the server to test the notification endpoint that is provided by the client, e.g. during subscription.

This method shall follow the provisions specified in the tables 8.4.7.3.2-1 and 8.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.7.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	Data type	Cardinality	Response Codes	Description	
	n/a		204 No Content	The notification endpoint was tested successfully. The response body shall be empty.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

8.4.7.3.3   PUT

Not supported.

8.4.7.3.4   PATCH

Not supported.

8.4.7.3.5   DELETE

Not supported.

8.5   Data model

8.5.1   Introduction

This clause defines the request and response data structures of the VNF Indicator interface.

8.5.2   Resource and notification data types

8.5.2.1   Introduction

This clause defines the data structures to be used in resource representations and notifications. If a request or response contains attributes not defined in the present specification, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

8.5.2.2   Type: VnfIndicator

This type represents a VNF indicator value. It shall comply with the provisions defined in table 8.5.2.2-1.

Table 8.5.2.2-1: Definition of the VnfIndicator data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of this VNF indicator.
name	String	0..1	Human readable name of the indicator. Shall be present if defined in the VNFD.
value	Object	1	Provides the value of the indicator. The value format is defined in the VNFD. See note.
vnfInstanceId	Identifier	1	Identifier of the VNF instance which provides the indicator value.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.
>vnfInstance	Link	1	Link to the related VNF instance resource.
NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			

8.5.2.3   Type: VnfIndicatorSubscriptionRequest

This type represents a subscription request related to VNF indicator value change notifications. It shall comply with the provisions defined in table 8.5.2.3-1.

Table 8.5.2.3-1: Definition of the VnfIndicatorSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	VnfIndicatorNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
authentication	SubscriptionAuthentication	0..1	Authentication parameters to configure the use of Authorization when sending notifications corresponding to this subscription, as defined in clause 4.5.3.4. This attribute shall only be present if the subscriber requires authorization of notifications.

8.5.2.4   Type: VnfIndicatorSubscription

This type represents a subscription related to notifications about VNF indicator value changes. It shall comply with the provisions defined in table 8.5.2.4-1.

Table 8.5.2.4-1: Definition of the VnfIndicatorSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	VnfIndicatorNotificationsFilter	0..1	Filter settings for this subscription, to define the subset of all notifications this subscription relates to. A particular notification is sent to the subscriber if the filter matches, or if there is no filter.
callbackUri	Uri	1	The URI of the endpoint to send the notification to.
_links	Structure (inlined)	1	Links for this resource.
>self	Link	1	URI of this resource.

8.5.2.5   Type: VnfIndicatorValueChangeNotification

This type represents a VNF indicator value change notification. It shall comply with the provisions defined in table 8.5.2.5-1.

The notification shall be triggered by the consumer when the value of an indicator has changed.

Table 8.5.2.5-1: Definition of the VnfIndicatorValueChangeNotification data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this notification. If a notification is sent multiple times due to multiple subscriptions, the "id" attribute of all these notifications shall have the same value.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "VnfIndicatorValueChangeNotification" for this notification type.
subscriptionId	Identifier	1	Identifier of the subscription that this notification relates to.
timeStamp	DateTime	1	Date-time of the generation of the notification.
vnfIndicatorId	IdentifierInVnfd	1	Identifier of the VNF indicator whose value has changed.
name	String	0..1	Human readable name of the VNF indicator. Shall be present if defined in the VNFD.
value	Object	1	Provides the value of the VNF indicator. The value format is defined in the VNFD. See note.
vnfInstanceId	Identifier	1	Identifier of the VNF instance which provides the indicator value.
_links	Structure (inlined)	1	Links for this resource.
>vnfInstance	Link	1	Link to the related VNF instance resource.
>subscription	Link	1	Link to the related subscription.
NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			

8.5.3   Referenced structured data types

8.5.3.1   Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

8.5.3.2   Type: VnfIndicatorNotificationsFilter

This type represents a subscription filter related to notifications about VNF indicator value changes. It shall comply with the provisions defined in table 8.5.3.1-1.

At a particular nesting level in the filter structure, the following applies: All attributes shall match in order for the filter to match (logical "and" between different filter attributes). If an attribute is an array, the attribute shall match if at least one of the values in the array matches (logical "or" between the values of one filter attribute).

Table 8.5.3.1-1: Definition of the VnfIndicatorNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
vnfInstanceSubscriptionFilter	VnfInstanceSubscriptionFilter	0..1	Filter criteria to select VNF instances about which to notify. See note.
indicatorIds	IdentifierInVnfd	0..N	Match particular VNF indicator identifiers.
NOTE: This attribute shall not be included when the VNFM sends a subscription request to a particular VNF instance.			

8.5.4 Referenced simple data types and enumerations

No particular simple data types and enumerations are defined for this interface, in addition to those defined in clause 4.4.

9 VNF Configuration interface

9.1 Description

This interface allows the VNFM to set configuration of a VNF instance and/or its VNFC instance(s).

The support of the VNF Configuration interface is optional.

The operation provided through this interface is:

- Set Configuration

9.2 Resource structure and methods

All resource URIs of the API shall use the base URI specification defined in clause 4.2. The string "vnfconfig" shall be used to represent {apiName}. The {apiVersion} shall be set to "v1" for the present document. All resource URIs in the clauses below are defined relative to the above base URI.

Figure 9.2-1 shows the overall resource URI structure defined for the VNF configuration interface.
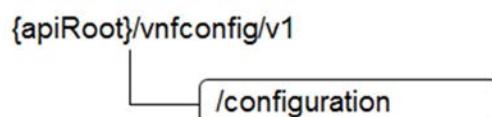
Figure 9.2-1: Resource URI structure of VNF configuration Interface

Table 9.2-1 lists the individual resources defined, and the applicable HTTP methods.

Table 9.2-1: Resources and methods overview of the VNF configuration interface

Resource name	Resource URI	HTTP Method	Meaning
Configuration	/configuration	PATCH	Set configuration data of a VNF instance and/or its VNFC instances.
		GET	Read configuration data of a VNF instance and its VNFC instances.

9.3   Sequence diagrams (informative)

9.3.1   Flow of setting the VNF configuration

This clause describes the procedure for setting the configuration of a VNF instance and/or its VNFC instances.
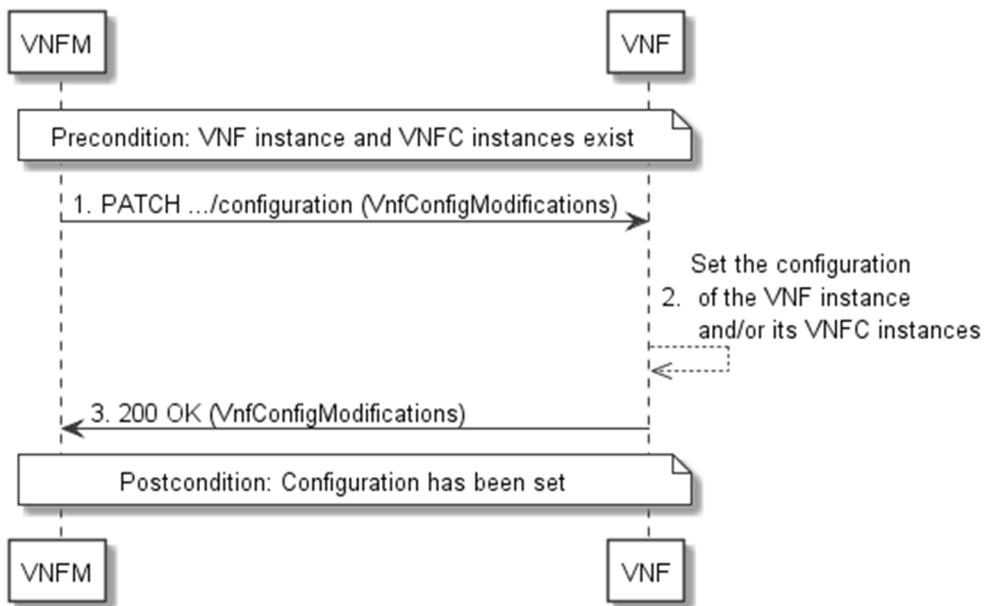
Figure 9.3.1-1: Flow of setting the configuration of a VNF instance and/or its VNFC instances

The procedure consists of the following steps as illustrated in figure 9.3.1-1.

Precondition: A VNF instance and its VNFC instances exist.

1.   The VNFM sends a PATCH request to the "configuration" resource including in the payload body a data structure of type "VnfConfigModifications".

2.   The VNF sets the configuration of the VNF instance and/or its VNFC instances.

3.   The VNF returns a "200 OK" response to the VNFM, and includes a data structure of type "VnfConfigModifications" in the payload body.

**Postcondition:** Configuration of the VNF instance and/or its VNFC instances has been set.

9.4   Resources

9.4.1   Introduction

This clause defines all the resources and methods provided by the VNF configuration interface.

9.4.2   Resource: Configuration

9.4.2.1   Description

This resource represents the configuration of a VNF instance and its VNFC instances. The client can use this resource to set and read the configuration of a VNF instance and its VNFC instances.

9.4.2.2   Resource definition

The resource URI is:

{apiRoot}/vnfconfig/v1/configuration

This resource shall support the resource URI variables defined in table 9.4.2.2-1.

Table 9.4.2.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

9.4.2.3   Resource methods

9.4.2.3.1   POST

Not supported.

9.4.2.3.2   GET

The client can use this method to read configuration information about a VNF instance and/or its VNFC instances.

This method shall follow the provisions specified in the tables 9.4.2.3.2-1 and 9.4.2.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.2.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.3.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Description		
	n/a				
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**	
	VnfConfiguration	1	200 OK	Configuration information about a VNF instance was read successfully. The response body shall contain a representation of the configuration resource, as defined in clause 9.5.3.2.	
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.	

9.4.2.3.3   PUT

Not supported.

9.4.2.3.4   PATCH

This method sets or modifies a configuration resource.

This method shall follow the provisions specified in the tables 9.4.2.3.4-1 and 9.4.2.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.2.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.2.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality		Description
Request body	VnfConfigModifications	1		The parameter for the configuration modification, as defined in clause 9.5.2.2.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfConfigModifications	1	200 OK	The request was accepted and completed. The response body shall contain the parameters of the configuration modification that was applied to the configuration resource (see clause 9.5.2.2).
	ProblemDetails	0..1	412 Precondition failed	Error: A precondition given in an HTTP request header is not fulfilled. Typically, this is due to an ETag mismatch, indicating that the resource was modified by another entity. The response body should contain a ProblemDetails structure, in which the "detail" attribute should convey more information about the error.
	ProblemDetails	See clauses 4.3.5.4/ 4.3.5.5	4xx/5xx	In addition to the response codes defined above, any common error response code as defined in clause 4.3.5.4, and any other valid HTTP error response as defined in clause 4.3.5.5, may be returned.

9.4.2.3.5   DELETE

Not supported.

9.5   Data model

9.5.1   Introduction

This clause defines the request and response data structures of the VNF Configuration interface. If a request or response contains attributes not defined in the present specification, a receiving functional block that does not understand these attributes shall not treat their presence as an error, and may choose to ignore them.

9.5.2   Resource and notification data types

9.5.2.1   Introduction

This clause defines the data structures to be used in resource representations and notifications.

9.5.2.2   Type: VnfConfigModifications

This type represents request parameters for the "Set Configuration" operation. It shall comply with the provisions defined in table 9.5.2.2-1.

Table 9.5.2.2-1: Definition of the VnfConfigModifications data type

Attribute name	Data type	Cardinality	Description
vnfConfigurationData	VnfConfigurationData	0..1	Modifications to configuration data for the VNF instance. See note 1. If present, the modifications of the "vnfConfigurationData" attribute shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).
vnfcConfigurationData	VnfcConfigurationData	0..N	Modifications to configuration data for certain VNFC instances. See note 1 and note 2. If present, the modifications of the "vnfcConfigurationData" attribute shall follow the provisions defined below this table.
NOTE 1: At least one of "vnfConfigurationData" and "vnfcConfigurationData" shall be present.			
NOTE 2: The VnfcConfiguration data type can only be used to modify the configuration of existing VNFC instances.			

The following provisions shall apply when modifying an attribute that is an array of objects of type "VnfcConfigurationData".

Assumptions:

1)   "oldList" is the "VnfcConfigurationData" array to be modified and "newList" is the "VnfcConfigurationData" array that contains the changes.

2)   "oldEntry" is an entry in "oldList" and "newEntry" is an entry in "newList".

3)   A "newEntry" has a "corresponding entry" if there exists an "oldEntry" that has the same content of the "vnfcInstanceId" attribute as the "newEntry"; a "newEntry" has no corresponding entry if no such "oldEntry" exists.

4)   In any array of "VnfcConfigurationData" structures, the content of "vnfcInstanceId" is unique (i.e. there shall be no two entries with the same content of "vnfcInstanceId").

Provisions:

1)   For each "newEntry" in "newList" that has no corresponding entry in "oldList", the "oldList" array shall be modified by adding that "newEntry".

2)   For each "newEntry" in "newList" that has a corresponding "oldEntry" in "oldList", the value of "oldEntry" shall be updated with the value of "newEntry" according to the rules of JSON Merge PATCH (see IETF RFC 7396 [15]).

9.5.3   Referenced structured data types

9.5.3.1   Introduction

This clause defines data structures that can be referenced from data structures defined in the previous clauses, but can neither be resource representations nor bound to any subscribe/notify mechanism.

9.5.3.2   Type: VnfConfiguration

This type represents configuration parameters of a VNF instance and its VNFC instances. It shall comply with the provisions defined in table 9.5.3.2-1.

Table 9.5.3.2-1: Definition of the VnfConfiguration data type

Attribute name	Data type	Cardinality	Description
vnfConfigurationData	VnfConfigurationData	1	Configuration parameters of the VNF instance
vnfcConfigurationData	VnfcConfigurationData	0..N	Configuration parameters of the VNFC instances

9.5.3.3   Type: VnfConfigurationData

This type represents configuration parameters of a VNF instance. It shall comply with the provisions defined in table 9.5.3.3-1.

Table 9.5.3.3-1: Definition of the VnfConfigurationData data type

Attribute name	Data type	Cardinality	Description
extCpConfig	CpConfiguration	0..N	Configuration parameters for the external CPs of the VNF instance.
dhcpServer	IpAddress	0..1	IP address of the DHCP server that the VNF instance can use to obtain IP addresses to be assigned to its external CPs.
vnfSpecificData	KeyValuePairs	0..1	Additional configurable properties of the VNF instance declared in the VNFD as "VnfConfigurableProperties". See note.
NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			

9.5.3.4   Type: VnfcConfigurationData

This type represents configuration parameters of a VNFC instance. It shall comply with the provisions defined in table 9.5.3.4-1.

Table 9.5.3.4-1: Definition of the VnfcConfigurationData data type

Attribute name	Data type	Cardinality	Description
vnfcInstanceId	IdentifierInVnf	1	Identifier of a VNFC instance to which this set of configuration data applies.
intCpConfig	CpConfiguration	0..N	Configuration parameters for the internal CPs of the VNFC instance.
dhcpServer	IpAddress	0..1	IP address of the DHCP server that the VNF can use to obtain IP addresses to be assigned to its CPs.
vnfcSpecificData	KeyValuePairs	0..1	Additional configurable properties of the VNFC instance declared in the VNFD as "VnfcConfigurableProperties". See note.
NOTE: ETSI GS NFV-SOL 001 [i.5] specifies the structure and format of the VNFD based on TOSCA specifications.			

9.5.3.5   Type: CpConfiguration

This type represents configuration parameters of a CP instance. It shall comply with the provisions defined in table 9.5.3.5-1.

Table 9.5.3.5-1: Definition of the CpConfiguration data type

Attribute name	Data type	Cardinality	Description
cpId	IdentifierInVnf	1	Identifier of a CP instance within the namespace of a specific VNF instance or a VNFC instance
cpdId	IdentifierInVnfd	1	Identifier of the CPD in the VNFD
addresses	CpAddress	1..N	Network address and port assigned to the CP

9.5.3.6 Type: CpAddress

This type represents configuration parameters of a CP instance address. It shall comply with the provisions defined in table 9.5.3.6-1.

Table 9.5.3.6-1: Definition of the CpAddress data type

Attribute name	Data type	Cardinality	Description
address	Structure (inlined)	0..1	Network address that has been configured on the CP. See note 1.
>macAddress	MacAddress	0..1	Mac address. See note 2.
>ipAddress	IpAddress	0..1	IP address. See note 2.
useDynamicAddress	Boolean	0..1	Set to true if an address shall be assigned dynamically. Otherwise set to false. The default value shall be false. See note 1.
port	UnsignedInt	0..1	The port assigned to the CP instance (e.g. IP port number, Ethernet port number, etc.).
NOTE 1: Either "address" or "useDynamicAddress" shall be present. NOTE 2: At least one of "macAddress" and "ipAddress" shall be present.			

9.5.4 Referenced simple data types and enumerations

No particular simple data types and enumerations are defined for this interface, in addition to those defined in clause 4.4.

Annex A (informative):
Mapping operations to protocol elements

A.1   Overview

This annex provides the mapping between operations as defined in ETSI GS NFV-IFA 008 [1] and the corresponding resources and HTTP methods defined in the present document.

A.2   VNF Lifecycle Management interface

ETSI ETSI GS NFV-IFA 008 [1] operation	HTTP method	Resource	Direction
Create VNF identifier	POST	vnflcm/v1/vnf_instances	EM → VNFM
Instantiate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/instantiate	EM → VNFM
Scale VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/scale	EM → VNFM VNF → VNFM
Scale VNF to Level	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/scale_to_level	EM → VNFM VNF → VNFM
Change VNF Flavour	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/change_flavour	EM → VNFM
Terminate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/terminate	EM → VNFM
Delete VNF identifier	DELETE	vnflcm/v1/vnf_instances/{vnfInstanceId}	EM → VNFM
Query VNF	GET	vnflcm/v1/vnf_instances/{vnfInstanceId}	EM → VNFM VNF → VNFM
	GET	vnflcm/v1/vnf_instances	EM → VNFM VNF → VNFM
Heal VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/heal	EM → VNFM VNF → VNFM
Operate VNF	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/operate	EM → VNFM
Change external VNF Connectivity	POST	vnflcm/v1/vnf_instances/{vnfInstanceId}/change_ext_conn	EM → VNFM
Modify VNF information	PATCH	vnflcm/v1/vnf_instances/{vnfInstanceId}	EM → VNFM
Get Operation Status	GET	vnflcm/v1/vnf_lcm_op_occs	EM → VNFM VNF → VNFM
	GET	vnflcm/v1/vnf_lcm_op_occs/{vnfLcmOpOccId}	EM → VNFM VNF→ VNFM
Subscribe	POST	vnflcm/v1/subscriptions	EM → VNFM VNF → VNFM
Query Subscription Information	GET	vnflcm/v1/subscriptions	EM → VNFM VNF → VNFM
	GET	vnflcm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF → VNFM
Terminate subscription	DELETE	vnflcm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF → VNFM
Notify	POST	(client-provided)	VNFM → EM VNFM → VNF

A.3 VNF Performance Management interface

ETSI GS NFV-IFA 008 [1] operation	HTTP method	Resource	Direction
Create PM Job	POST	vnfpm/v1/pm_jobs	EM → VNFM
Delete PM Job	DELETE	vnfpm/v1/pm_jobs/{pmJobId}	EM → VNFM
Query PM Job	GET	vnfpm/v1/pm_jobs	EM → VNFM
	GET	vnfpm/v1/pm_jobs/{pmJobId}	EM → VNFM
Create Threshold	POST	vnfpm/v1/thresholds	EM → VNFM
Delete Thresholds	DELETE	vnfpm/v1/thresholds/{thresholdId}	EM → VNFM
Query Threshold	GET	vnfpm/v1/thresholds	EM → VNFM
	GET	vnfpm/v1/thresholds/{thresholdId}	EM → VNFM
Subscribe	POST	vnfpm/v1/subscriptions	EM → VNFM VNF→ VNFM
Query Subscription Information	GET	vnfpm/v1/subscriptions	EM → VNFM VNF→ VNFM
	GET	vnfpm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF→ VNFM
Terminate subscription	DELETE	vnfpm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF→ VNFM
Notify	POST	(client-provided)	VNFM → EM VNFM → VNF

A.4 VNF Fault Management interface

ETSI GS NFV-IFA 008 [1] operation	HTTP method	Resource	Direction
Get Alarm List	GET	vnffm/v1/alarms	EM → VNFM VNF → VNFM
Acknowledge Alarm	PATCH	vnffm/v1/alarms/{larmId}	EM → VNFM VNF → VNFM
Escalate Perceived Severity	POST	vnffm/v1/alarms/{alarmId}/escalate	EM → VNFM VNF → VNFM
Subscribe	POST	vnffm/v1/subscriptions	EM → VNFM VNF → VNFM
Query Subscription Information	GET	vnffm/v1/subscriptions	EM → VNFM VNF → VNFM
	GET	vnffm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF → VNFM
Terminate subscription	DELETE	vnffm/v1/subscriptions/{subscriptionId}	EM → VNFM VNF → VNFM
Notify	POST	(client-provided)	VNFM → EM VNFM → VNF

A.5   VNF Indicator interface

ETSI GS NFV-IFA 008 [1] operation	HTTP method	Resource	Direction
Get Indicator Value	GET	vnfind/v1/indicators	VNFM → EM VNFM → VNF
	GET	vnfind/v1/indicators/{vnfInstanceId}	VNFM → EM VNFM → VNF
	GET	vnfind/v1/indicators/{vnfInstanceId}/{indicatorId}	VNFM → EM VNFM → VNF
Subscribe	POST	vnfind/v1/subscriptions	VNFM → EM VNFM → VNF
Query Subscription Information	GET	vnfind/v1/subscriptions	VNFM → EM VNFM → VNF
	GET	vnfind/v1/subscriptions/{subscriptionId}	VNFM → EM VNFM → VNF
Terminate subscription	DELETE	vnfind/v1/subscriptions/{subscriptionId}	VNFM → EM VNFM → VNF
Notify	POST	(client-provided)	EM → VNFM VNF → VNFM

A.6   VNF Configuration interface

ETSI GS NFV-IFA 008 [1] operation	HTTP method	Resource	Direction
Set Configuration	PATCH	vnfconfig/v1/configuration	VNFM → VNF
	GET	vnfconfig/v1/configuration	VNFM → VNF

Annex B (informative):
Explanations

B.1   Introduction

This annex provides explanations of certain concepts introduced in the present document.

In clause B.2, the underlying concepts of scaling a VNF instance are explained.

In clause B.3, examples of VNF connectivity patterns and change of VNF external connectivity are provided.

B.2   Scaling of a VNF instance

A VNF instance can be scaled in the following ways:

- scale out: adding additional VNFC instances to the VNF to increase capacity

- scale in: removing VNFC instances from the VNF, in order to release unused capacity

This mechanism is called "horizontal scaling".

> NOTE: Besides that, there is also "vertical scaling" which is not supported in the present document, and which includes scale up (adding further resources to existing VNFC instances) and scale down (removing resources from existing VNFC instances).

Potentially, different *aspects* of a VNF can be scaled independently. For example, a VNF could be designed to provide static capacity such as database nodes and dynamic capacity such as query processing nodes. Such a VNF might be scaled with regards to two separate aspects: the "static capacity" aspect can be scaled by adding VNFCs from VNF Deployment Units (VDUs) defining database nodes, and the "dynamic capacity" aspect can be scaled by adding VNFCs from VDUs defining query processing nodes. In complex VNF designs, scaling a VNF often requires adding/removing a number of related VNFC instances of several different types, possibly based on multiple VDUs. For example, in a high availability configuration, it might be required to add in each scaling step a pair of VNFC instances, one in active and one in standby configuration. The scaling aspects valid for a particular VNF are declared in the VNFD.

Each scaling aspect can only be scaled in discrete steps, the so-called "*scaling steps*". Each scaling step corresponds to adding or removing an *increment* (set of VNFCs based on one or more VDUs, and the related virtualised storage/virtualised network resources) to or from the VNF instance, and (re)configuring the virtualised resources. Per increment, the VNFM will figure out the necessary set of VNFCs and the related set of resources based on VNF-specific rules, for instance using the lifecycle management script associated to the Scale VNF or Scale VNF to Level event.

When scaling a VNF for a particular aspect, the number of scaling steps to apply to that aspect can be provided as a parameter. A scaling step is the smallest unit by which a particular aspect of a VNF can be scaled, and is mapped by the VNFM to the addition (or removal) of a certain set of resources. For each scaling aspect, the minimum scale level is assumed as zero, and the maximum scale level is defined in the VNFD. The maximum scale level corresponds to the maximum number of scaling steps that can be performed for this aspect, starting from the minimum scale level (i.e. zero). The maximum scale level represents the maximum configuration of that aspect of the VNF in a given deployment flavour. The minimum scale level represents the minimum configuration of that aspect of the VNF in a given deployment flavour. It usually corresponds to some deployed resources, but it is also possible to define in the VNFD that certain VDUs may not always have a corresponding VNFC instance, i.e. for certain aspects the minimum configuration may indeed be empty.

At each point in time between the completed VNF instantiation and the VNF termination, the current "size" of a particular scaling aspect of the VNF can be expressed by the current scale level w.r.t. that aspect. When the VNF is instantiated, the current scale level is initialized with values that are defined as part of the instantiation level in the VNFD for the associated aspect. Figure B.2-1 illustrates the concepts described above.
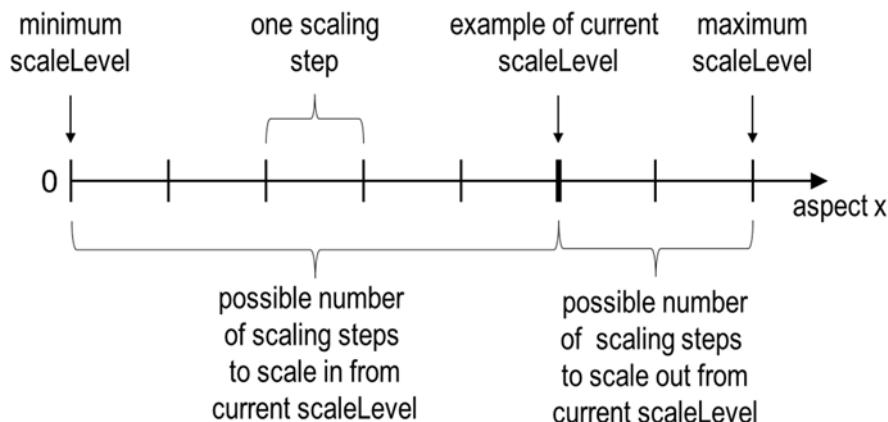
**Figure B.2-1: Illustrating the concepts of scale level and scaling steps
for a particular scaling aspect**

As indicated above, a VNF can have one or more scaling aspects. Each individual aspect has a current scale level. All pairs of (aspect, scaleLevel) together are called the *scale status* of the VNF instance and can be obtained from the "scaleStatus" attribute of the VnfInstance structure which is returned when reading the "Individual VNF instance" resource or when querying the "VNF instances" resource. Example 1 illustrates a possible scale status.

    EXAMPLE 1:

```
"scaleStatus": [
    {"aspectId": "processing", "scaleLevel": "2"},
    {"aspectId": "database", "scaleLevel": "3"}
]
```

When requesting scaling of a VNF instance, there are two methods: Scale VNF (see clause 5.4.5) and Scale VNF to Level (see clause 5.4.6). When using "Scale VNF", the scaling request defines how many increments (scaling steps) are requested to be added to or removed from the current "size" (scale level) *for a single aspect*. Depending on the VNF capabilities, single-step scaling or multiple-step scaling can be supported in a single scale request. When using "Scale VNF to Level", the scale request defines a target size of the VNF instance by defining the requested target size *for all aspects at once*, independent from the current scale status (current size) of the VNF instance. The target size can be expressed by referencing pre-defined sizes (called *instantiation levels*) declared in the VNFD, or by explicitly providing the target scale level for each scaling aspect, as illustrated in Example 2.

    EXAMPLE 2:

```
"scaleInfo": [
    {"aspectId": "processing", "scaleLevel": "4"},
    {"aspectId": "database", "scaleLevel": "2"}
]
```

These combinations allow four sub-modes of scaling:

- Scale VNF with a single step

- Scale VNF with multiple steps

- Scale VNF to Level based on pre-defined sizes (instantiation levels) only

- Scale VNF to Level with arbitrary sizes

B.3 Examples of VNF connectivity patterns

B.3.1 Introduction

Clause B.3.2 illustrates examples of possible connectivity patterns for a VNF. The purpose is to illustrate the relationship among the different information elements specified in clause 5.5 that are used to describe the connectivity of and within a VNF instance.

NOTE: The information related to connectivity as shown in clause B.3.2 is to be understood in the context of the present document, i.e. availability of certain information on the Ve-Vnfm reference point follows the conditions that are detailed in the respective attribute descriptions and notes in the present document.

Clause B.3.3 illustrates the use of the "Change external VNF connectivity" task resource to re-connect external CPs of a VNF instance to a different external VL.

B.3.2 Example of a VNF with two different types of external connection points

The present example shows a regular connectivity pattern of a VNF where the two external CPs of the VNF use different connectivity patterns. Figure B.3.2-1 illustrates the example, from which it is highlighted the following:

- An external CP of the VNF instance (see VnfExtCp #1) that maps to an internal CP, i.e. a CP of a specific VNFC.

- An external CP of the VNF instance (see VnfExtCp #2) that refers to a link port of an internal VL of the VNF (see VnfLinkPort #2.2).

- An internal VL of the VNF instance (see VnfVirtualLink #1) that is only used for connectivity of VNFCs within the VNF.

- An internal VL of the VNF instance (see VnfVirtualLink #2) that is used as provider of a link port for connectivity of external CPs of the VNF.

- Link ports of internal VL(s) of the VNF instance (see VnfLinkPort #1.1 to #1.3 and VnfLinkPort #2.1), which are exposed on Ve-Vnfm reference point.

- Internal CPs, i.e. CPs of specific VNFCs (see VNFC CPs), which are exposed on the Ve-Vnfm reference point.
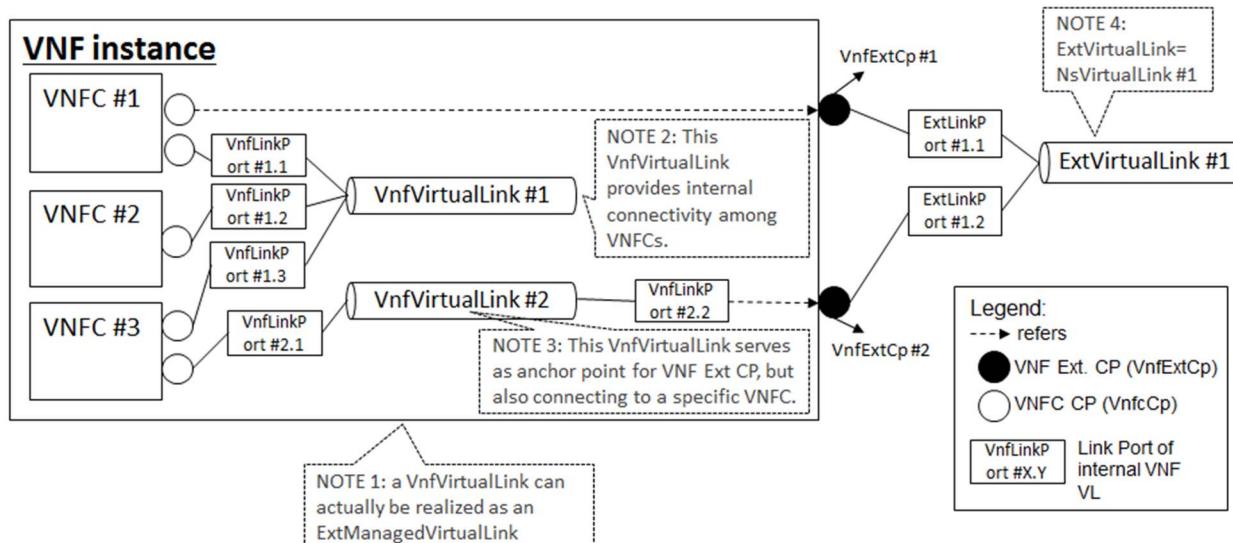

Figure B.3.2-1: Example of a VNF with two different types of external connections points

B.3.3   Example of changing VNF connectivity

This example illustrates changing the external connectivity of a VNF instance using the "Change external VNF connectivity" task resource (clause 5.4.11). The scenario depicted disconnects from a "source" external VL all those external CP instances that were created based on a particular CPD, and connects them to a "target" external VL.
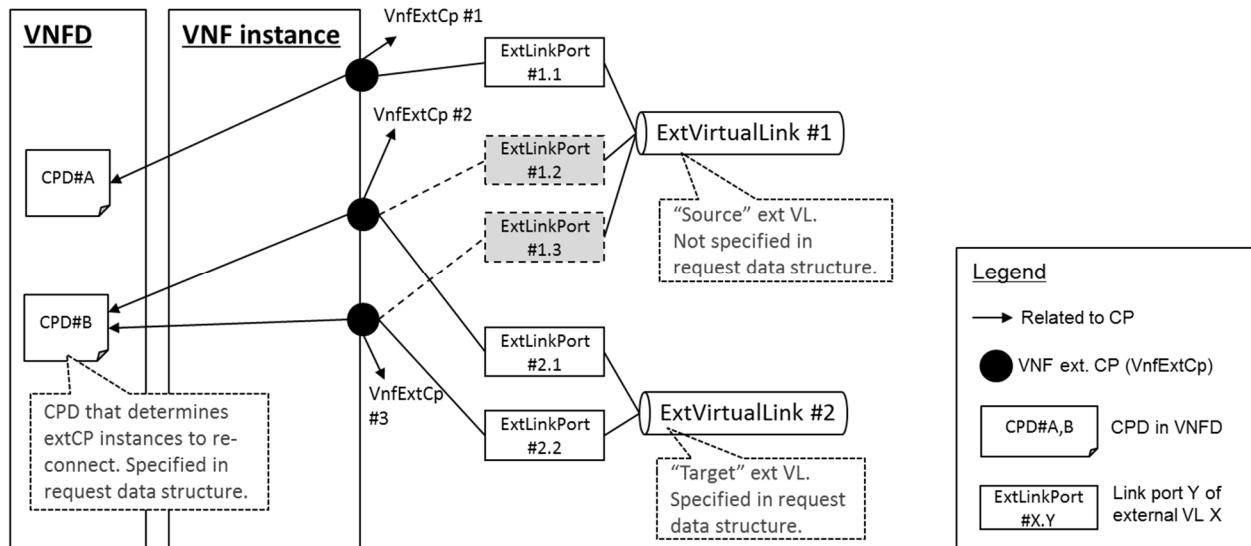
Figure B.3.3-1: Illustration of disconnecting external CPs from one external VL and connecting them to another external VL

Annex C (informative):
Differences between SOL002 and SOL003

C.1   Overview

The set of APIs defined in ETSI GS NFV-SOL 002 (the present document) and ETSI GS NFV-SOL 003 [i.2] are overlapping to a large extent. A number of APIs are present in both ETSI GS NFV-SOL 002 and ETSI GS NFV-SOL 003 [i.2]. However, for each of these APIs, there are certain differences depending on whether they are exposed towards the NFVO, or towards the VNF/EM. These differences are described in clause C.2.

Other APIs are only present in one of the two specifications, as they only make sense either on the Ve-Vnfm reference point, or on the Or-Vnfm reference point. These APIs are listed in clause C.3.

C.2   Interfaces present in both SOL002 and SOL003

C.2.1   Basic principles

When the NFVO requests VNF management functionality from the VNFM, it addresses each VNF instance as a whole. Detailed information about VNF internals (such as VNFC instances or internal topology) is typically not needed and cannot be managed; however, the NFVO needs to understand the resource view of the VNFCs in order to perform resource orchestration. In contrast, the entity actually performing the management of a VNF instance (the EM or the VNF instance itself) needs detailed information about VNF internals, such as VNFC instances or internal topology. This difference results in more detailed VNFC-related information to be exposed in SOL002 (as part of certain resource representations, notifications and request parameters) than in SOL003. Also, certain operation modes such as graceful termination are not needed by the EM (as opposed to the NFVO), as the EM can ensure to take a VNF instance out of service before requesting termination.

C.2.2   VNF Lifecycle Management interface

Certain attributes are only available on either SOL002 or SOL003, or have restrictions w.r.t. their value set. Some operations on certain resources are only available on one branch of Ve-Vnfm, i.e. either towards the VNF or towards the EM, or only available on either SOL002 or SOL003. Resources and attributes with such restrictions are documented in table C.2.2-1.

Table C.2.2-1: VNF Lifecycle Management interface - differences between SOL002 and SOL003

Attribute/Resource with method	SOL002	SO003
VnfInstance. instantiatedVnfInfo.vnfcInfo	present	not present
VnfInstance.vimConnectionInfo	not present	present
VnfInfoModifications.vnfcInfoModifications	present	not present
VnfInfoModifications.vimConnectionInfo	not present	present
InstantiateVnfRequest.vimConnectionInfo	not present	present
ChangeVnfFlavourRequest.vimConnectionInfo	not present	present
ChangeExtVnfConnectivityRequest.vimConnectionInfo	not present	present
VnfInfoModificationRequest.vimConnectionInfo	Not present	present
TerminateVnfRequest.terminationType	values: "FORCEFUL"	values: "FORCEFUL", "GRACEFUL"
TerminateVnfRequest.gracefulTerminationTimeout	not present	present
OperateVnfRequest.stopType	values: "FORCEFUL"	values: "FORCEFUL", "GRACEFUL"
OperateVnfRequest.gracefulStopTimeout	not present	present
HealVnfRequest.vnfcInstanceId	present	not present
HealVnfRequest.healScript	present	not present
OperateVnfRequest.vnfcInstanceId	present	not present
VnfcResourceInfo.vnfcCpInfo	mandatory	conditional
VnfVirtualLinkResourceInfo.vnfLinkPorts	mandatory	conditional
POST .../vnf_instances	available to EM only	available
POST .../vnf_instances/{vnfInstanceId}/instantiate	available to EM only	available
POST .../vnf_instances/{vnfInstanceId}/change_flavour	available to EM only	available
POST .../vnf_instances/{vnfInstanceId}/terminate	available to EM only	available
DELETE .../vnf_instances/{vnfInstanceId}	available to EM only	available
POST .../vnf_instances/{vnfInstanceId}/operate	available to EM only	available
POST .../vnf_instances/{vnfInstanceId}/change_ext_conn	available to EM only	available
PATCH .../vnf_instances/{vnfInstanceId}	available to EM only	available

C.2.3   VNF Performance Management interface

As opposed to SOL003, information about VNFCs can be added to the measurements in SOL002, if applicable to the actual measurement as defined in an external measurement specification.

Certain attributes are only available on either SOL002 or SOL003, or have restrictions w.r.t. their value set. Some operations on certain resources are only available on one branch of Ve-Vnfm, i.e. either towards the VNF or towards the EM, or only available on either SOL002 or SOL003. Resources and attributes with such restrictions are documented in table C.2.3-1.

Table C.2.3-1: VNF Performance Management interface - differences between SOL002 and SOL003

Attribute/Resource with method	SOL002	SO003
ThresholdCrossedNotification.subObjectInstanceId	present	not present
PerformanceInformationAvailableNotification. subObjectInstanceIds	present	not present
CreatePmJobRequest.subObjectInstanceIds	present	not present
PmJob.subObjectInstanceIds	present	not present
CreateThresholdRequest. subObjectInstanceIds	present	not present
Threshold.subObjectInstanceIds	present	not present
PerformanceReport.subObjectInstanceId	present	not present
POST .../pm_jobs	available to EM only	available
DELETE .../pm_jobs/{pmJobId}	available to EM only	available
GET .../pm_jobs	available to EM only	available
GET .../pm_jobs/{pmJobId}	available to EM only	available
POST .../thresholds	available to EM only	available
DELETE .../thresholds/{thresholdId}	available to EM only	available
GET .../thresholds	available to EM only	available
GET .../thresholds/{thresholdId}	available to EM only	available

C.2.4   VNF Fault Management interface

As opposed to SOL003, information about VNFCs affected by a fault is added to the alarms in SOL002. Also, it is possible for the EM to suggest escalation of the perceived severity of an alarm in SOL002.

Certain attributes are only available on either SOL002 or SOL003, or have restrictions w.r.t. their value set. Some operations on certain resources are only available on either SOL002 or SOL003. Resources and attributes with such restrictions are documented in table C.2.4-1.

Table C.2.4-1: VNF Fault Management interface - differences between SOL002 and SOL003

Attribute/Resource with method	SOL002	SOL003
Alarm.vnfcInstanceIds	available	not available
POST …/alarms/{alarmId}/escalate	available	not available

C.2.5   VNF Indicator interface

There are no differences between SOL002 and SOL003 w.r.t. the VNF Indicator interface. Notice that in case of SOL002, the consumer of this interface is the VNFM and the producer can be either EM or VNF.

C.3   Interfaces present in one of SOL002 and SOL003

C.3.1   Interfaces only present in SOL002

The following interfaces are only present in ETSI GS NFV-SOL 002 (the present document):

- VNF Configuration interface

C.3.2   Interfaces only present in SOL003

The following interfaces are only present in ETSI GS NFV-SOL 003 [i.2]:

- VNF Package Management interface

- VNF Lifecycle Operation Granting

- Virtualised Resources Quota Available Notification interface

Annex D (informative):
Authors & contributors

The following people have contributed to the present document:

Rapporteur:
Jong-Hwa Yi, ETRI

Other contributors:
Bruno Chatras, Orange

Uwe Rauschenbach, Nokia Networks

Anatoly Andrianov, Nokia

Ernest Bayha, Ericsson

Wooyong Choi, SK Telecom

Haibin Chu, Ericsson

Aijuan Feng, Huawei

Dmytro Gassanov, NetCracker

Junyi Jiang, Huawei

TaeYeon Kim, ETRI

Yuuya Kuno, DOCOMO Communications Lab.

René Robert, Orange

Kazuaki Obana, DOCOMO Communications Lab.

Myung-Ki Shin, ETRI

Xu Yang, Huawei

Yuya Kuno, DOCOMO Communications Lab.

Szabolcs Deak, Nokia

Lars-Erik Helander, Procera Networks

Hunor Demeter, Nokia

Annex E (informative):
Change History

Version	Date	Information about changes
v0.0.1	May 2016	Skeleton and scope based on: • NFVSOL(16)000003r1_SOL002_-_Proposed_table_of_contents • NFVSOL(16)000004_SOL002_-_Proposed_scope
v0.0.2	May 2016	Implemented NFVSOL(16)000017r1
v0.0.3	April 2017	Contributions incorporated • NFVSOL(17)000073 Document structure of SOL002 • NFVSOL(17)000073 Document structure of SOL002 • NFVSOL(17)000102r1 SOL002 SOL003 Remove Annex C resp X • NFVSOL(17)000161 SOL002- 2. References • NFVSOL(17)000147r1 SOL002 SOL003 Definitions, symbols and abbreviations • NFVSOL(17)000162r1 SOL002-clause 4 General aspects • NFVSOL(17)000131r3 SOL002-VNFLCM interface based on SOL003 • NFVSOL(17)00082r3 SOL002 Additional parameters of VNF LCM • NFVSOL(17)000133r1 SOL002-vnfInstance data type • NFVSOL(17)000132r1 SOL002-VNF Indicator interface based on SOL003 • NFVSOL(17)000163 SOL002-Annex B. Mapping operations • NFVSOL(17)000121_Conventions_three_parts_of_remarks_column • NFVSOL(17)000187r1_SOL002_SOL003_Conventions_global_fix_for_normative_statement • NFVSOL(17)000150r1 SOL002 SOL003 Adding description of rollback retry cancel fail to clause 5.4.13.1 • NFVSOL(17)000149 SOL002 SOL003 LCM ed note error handling bugfix • NFVSOL(17)000125r1 SOL002 SOL003 MonitoringParameters data structure • NFVSOL(17)000141 SOL002 SOL003 Rename ind to vnfind • NFVSOL(17)000157 SOL003 SOL002 5.2.1 Fixing Vnf Instance Creation flow • NFVSOL(17)000155r1 SOL003 SOL002 resolve Auto-X editor s note • NFVSOL(17)000151r1 SOL003 SOL002 clause 5.6.1 Basic concepts (for LCM errors) • NFVSOL(17)000126r1 SOL002 SOL003 NetworkAddress data structure • NFVSOL(17)000182 SOL002 SOL003 Indicators clean up NFVSOL(17)000188 SOL002 SOL003 Notification id NFVSOL(17)000189r1 SOL002 SOL003 VnfInstanceSubscriptionFilter general data type • NFVSOL(17)000190 SOL002 SOL003 VnfLcOpOcc fixes for ModifyVnfInfo • NFVSOL(17)000191r1 SOL002 SOL003 state change timestamp and affected resources • NFVSOL(17)000192 SOL002 SOL003 Remove editor s note in 5.4.3.3.4 • NFVSOL(17)000199_SOL002_SOL003_Renaming_attribute_selectors • NFVSOL(17)000200_SOL002_SOL003_Attribute_filter_equality • NFVSOL(17)000078r3_SOL002_SOL003_VimId_fixes • NFVSOL(17)000209r1_SOL002_SOL003_SOL005_all_fields__and_defaults_for_selectors (1) • NFVSOL(17)000221r1_SOL002_-_Data_model_of_VNF_indicator_interface_based_on_SOL0 • NFVSOL(17)000222r1_SOL002_-_VNF_FM_interface_based_on_SOL003 • NFVSOL(17)000223r1_SOL002_-_VNF_PM_interface_based_on_SOL003

Version	Date	Information about changes
v0.1.0	May 2017	• NFVSOL(17)000373r2 SOL002 - Mirror for Authorization • NFVSOL(17)000380r1 SOL002 - Mirror for clause 5 • NFVSOL(17)000381 SOL002 - Mirror for clause 6 • NFVSOL(17)000219R4 SOL002: VNF FM Interface - New Escalate Perceived Severity Operation • NFVSOL(17)000331r2 SOL002 B.2 and 5.1: Update Mapping Operations to Protocol for Scale VNF to Level • NFVSOL(17)000368 SOL002 SOL003 4.3.3.1 adding informative to overview and example consistently • NFVSOL(17)000354r3 SOL003 SOL002 autoscale autoheal description • NFVSOL(17)000349R2 SOL003 SOL002 5.x Error code 404 if task resource not supported • NFVSOL(17)000384 SOL003 SOL002 Refactoring VNF Instance link in AlarmNotification • NFVSOL(17)000375r1 SOL003 SOL002 meaning of OperateVnf • NFVSOL(17)000374 SOL003 SOL002 Notification Authorization future proofing • NFVSOL(17)000394_SOL002_SOL003_address_comments_from_Procera_Net works • NFVSOL(17)000393 SOL002 SOL003 Move VimConnectionInfo to the correct clause • NFVSOL(17)000392r1 SOL003 SOL002 missing notification triggers • NFVSOL(17)000343 SOL002 SOL003 global Renaming of attribute filters to attribute-based filtering • NFVSOL(17)000363 SOL002 SOL003 global consistency of enum type names • NFVSOL(17)000396r2 SL002 SOL003 6.4 7.4 8.4 9.4 10.4 11.4 Addition of the note about how to retrieve the resource id • NFVSOL(17)000397 SOL002 SOL003 5.3.3 Clarification of notification flow in the figure 5.3.3.1-1 • NFVSOL(17)000399 SOL003 SOL002 changedExtVLs in LcmOpOccNotif
v0.2.0	June 2017	• NFVSOL(17)000336r1 SOL003 many Fixing some conditions • NFVSOL(17)000282_SOL003_-_Removing_normative_dependencies_on_SOL001 • NFVSOL(17)000341_SOL003_7_5_2_5_Refactoring_links_in_AlarmNotification • NFVSOL(17)000217r4_SOL003_VNF_FM_Acknowledge_Alarm_operation • NFVSOL(17)000227r1_SOL003_4_3_2_2_EN_Attribute_filters_point_in_time • NFVSOL(17)000232_SOL003_5_5_3_7_rapp_note__information_element • NFVSOL(17)000280r2_SOL003_-_Clause_4_4_2_-_5_5_3_-_MAC_and_IP_address_represent • NFVSOL(17)000293r1_SOL003_4_2_Consistency_of_URI_and_OAuth • NFVSOL(17)000298r2_SOL003_5_4_3_2_Improvement_of_resource_definition_descriptioon • NFVSOL(17)000299r1_SOL003_5_4_13_2_Improvement_of_resource_definition_description • NFVSOL(17)000300_SOL003_4_4_2_Clear_meaning_of_IdentifierLocal_type • NFVSOL(17)000301r1_SOL003_5_4_16_Add_the_supplement_to_Finally_Failed • NFVSOL(17)000302r1_SOL003_4_3_5_5_Consistency_between_4_3_5_4_and_4_3_5_5 • NFVSOL(17)000303r3_SOL003_-_Editorial_changes • NFVSOL(17)000306r1_SOL003_4_3_2_2_Filter_Spec_Fix • NFVSOL(17)000307r2_SOL003_4_3_4_3_Adding_retry-after_header_field • NFVSOL(17)000335r1_SOL003_4_3_3_2_1_Fixes_to_attribute_selector • NFVSOL(17)000355_SOL003_SOL002_Replace_entity_body_by_payload_body
v0.3.0	June 2017	• NFVSOL(17)000423 SOL002 ExtCP ExtVL fixes related to IFA discussion • NFVSOL(17)000429r1 SOL002 Implementing mirror contributions based on SOL003 • NFVSOL(17)000020r3 - SOL002 REST based Resource Design for VNF Configuration Interface

Version	Date	Information about changes
v0.4.0	June 2017	• NFVSOL(17)000471 - SOL002 Resolving differences between SOL002 and SOL003 • NFVSOL(17)000470 - SOL002 FaultyResourceInfo data type in clause 7.5.3.3 • NFVSOL(17)000469 - SOL002 Modifications on clause 6.4.5.2 • NFVSOL(17)000468 - SOL002 Implementing mirror contributions of SOL003 • NFVSOL(17)000472r2 - SOL002 Clause 9. VNF Configuration • NFVSOL(17)000021r5 - SOL002 - VNF configuration flows • NFVSOL(17)000444R1_Summary_of_differences_between_SOL002___SOL003
v0.5.0	July 2017	• NFVSOL(17)000480r3_SOL002_Alignment_with_SOL003_general_clauses_1_to_4 • NFVSOL(17)000482r1_SOL002_Annex_to_compare_SOL002_and_SOL003 (1) • NFVSOL(17)000484_SOL002_VNFC_configuration_management_in_ModifyVnfInfo • NFVSOL(17)000442r6_SOL002_Data_structure_for_VNF_configuration • NFVSOL(17)000479_SOL002_VimConnectionInfo_removal_aligned_with_IFA008 • NFVSOL(17)000481r7_SOL002_Alignment_with_SOL003_clause_5_LCM_interface • NFVSOL(17)000483r2_SOL002_Fixing_clauses_6_and_7_PM_and_FM_interfaces • NFVSOL(17)000485r1_SOL002_Fixing_clause_8_Indicator_interface • NFVSOL(17)000486_SOL002_Alignment_with_SOL003_-_Annexes • NFVSOL(17)000487r2_SOL002_-_GET_method_for_VNF_Configuration • NFVSOL(17)000489r1_SOL002_handling_of_VNFC_in_FM_and_PM_interfaces • NFVSOL(17)000490_SOL002_2_1_4_3_Consistency_of_Range_Requests • NFVSOL(17)000491_SOL002_-_Annex_A_and_Editors_notes
v0.5.1	July 2017	• Implement additional editorial review comments received from SOL email list.
v0.6.0	July 2017	• NFVSOL(17)000503r1_SOL002_bug_vnfcInfo_wrong_level
v2.3.2	November 2017	• NFVSOL(17)000565_SOL002_SOL003_Fixing_actors_in_authorization_flows • NFVSOL(17)000589_SOL002_handling_of_VNFC_in_FM_interface_aligned_with_stage_2 • NFVSOL(17)000590r1_SOL002_handling_of_VNFC_in_PM_interface_aligned_with_stage_2 • NFVSOL(17)000593r1_SOL002_SOL003_miscellaneous_small_bugfixes.docx • NFVSOL(17)000622r2_SOL002_miscellaneuos_bugfixes

Version	Date	Information about changes
V2.3.3	December 2017	• NFVSOL(17)000644r1 SOL002 remove graceful mode from OperateVnf • NFVSOL(17)000636 SOL002 VNFC CP changes in AffectedVnfc • NFVSOL(17)000667r2 SOL002 SOL003 Add description to VNF fault management interface • NFVSOL(17)000668 SOL002 SOL003 complement the description of CancelModeType • NFVSOL(17)000691r1 SOL002/003 - Use of verbal forms for the expression of provisions • NFVSOL(17)000646 SOL002 SOL003 Add resource metadata to AffectedVnfc/VirtualLink/Storage • NFVSOL(17)000670r2 SOL002 SOL003 Fixing statement for mandatory and conditional HTTP methods • NFVSOL(17)000722 SOL002 fixing VNF connectivity figure • NFVSOL(17)000669r3 SOL002 SOL003 Add cancel time out attribute • NFVSOL(17)000635r1 SOL002 SOL003 implicit changes in VnfInfoModifications • NFVSOL(17)000698 SOL002 SOL003 allow Fail operation in FAILED_TEMP • NFVSOL(17)000715r2 SOL002 SOL003 - Double Subscriptions for Notifications • NFVSOL(17)000718r2 SOL002 SOL003 sequence of requests responses and notifications • NFVSOL(17)000388r6 SOL002 ed2.4.1 and SOL003 ed 2.4.1: Autorization of API Requests and Notifications Document type should be "CR" • NFVSOL(17)000674r4 SOL002 SOL003 Authorization method negotiation • NFVSOL(17)000695r1 SOL003 Fixing normative status of notification endpoint • NFVSOL(17)000754_SOL002_SOL003_Remove_redundant_description_of_vnf Configurabl • NFVSOL(17)000764 SOL002 align normative statements in trigger conditions mirror 734 • NFVSOL(17)000758r1 SOL002 Add clarification of ExManagedVirtualLink and ExtVirtualLink • NFVSOL(17)000671r2 SOL002/SOL003 ExtCpData changes from IFA1029r2 • NFVSOL(17)000780r1_SOL003_Fixing_leftovers_of_onboardedVnfPkg_Info_Id • NFVSOL(17)000789r2_Additional_modifications_for_SOL002_and_SOL003
V2.3.4	February 2008	Following comments raised during the approval period, it was decided to REMOVE all normative references to the IFA027 draft is this version of SOL002.

History

Document history		
V2.3.1	August 2017	Publication
V2.4.1	February 2018	Publication