ETSI GS NFV-SOL 005 V2.5.1 (2018-09)

**Network Functions Virtualisation (NFV) Release 2;
Protocols and Data Models;
RESTful protocols specification for
the Os-Ma-nfvo Reference Point**

Reference

RGS/NFV-SOL005ed251

Keywords

API, NFV, protocol

ETSI

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

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

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

Contents

Intellectual Property Rights

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 protocol specifications and data models fulfilling the requirements specified in ETSI GS NFV-IFA 013 [3] for the interfaces used over the Os-Ma-Nfvo 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 https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] IANA: "Assigned Internet Protocol Numbers".

NOTE: Available at https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.

[2] ETSI GS NFV-IFA 010: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Functional requirements Specification".

[3] ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

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

[5] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification".

[6] IEEE 802.1Q-2014: "IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks".

[7] IETF RFC 791: "Internet Protocol".

NOTE: Available at https://tools.ietf.org/html/rfc791.

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

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

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

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

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

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

[11] IETF RFC 4291: "IP Version 6 Addressing Architecture".

NOTE: Available at https://tools.ietf.org/html/rfc4291.

[12] IETF RFC 4632: "Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan".

NOTE: Available at https://tools.ietf.org/html/rfc4632.

[13] IETF RFC 4776: "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information".

NOTE: Available at https://tools.ietf.org/html/rfc4776.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[28] IETF RFC 8200: "Internet Protocol, Version 6 (IPv6) Specification".

NOTE: Available at https://tools.ietf.org/html/rfc8200.

[29] ISO 3166 (all parts): "Codes for the representation of names of countries and their subdivisions".

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

[31] ETSI GS NFV-IFA 027: "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Performance Measurements Specification".

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

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

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

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

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

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

[35] Semantic Versioning 2.0.0.

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

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

[i.2] Hypertext Transfer Protocol (HTTP) Status Code Registry at IANA.

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

[i.3] ETSI GS NFV-SOL 001: "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification".

[i.4] OpenStack: "Disk and container formats for images".

NOTE: Available from http://docs.openstack.org/image-guide/image-formats.html.

[i.5] OpenAPI Specification.

NOTE: Available at https://github.com/OAI/OpenAPI-Specification.

3 Abbreviations

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

API	Application Programming Interface
BSS	Business Support System
CIDR	Classless Inter-Domain Routing
CP	Connection Point
CPD	CP Descriptor

DF	Deployment Flavour
DSCP	Differentiated Services Code Point
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
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IFA	Interfaces and Architecture
IP	Internet Protocol
JSON	JavaScript Object Notation
LB	Load Balancing algorithm
LCCN	Lifecycle Change Notification
LCM	Lifecycle Management
MAC	Medium Access Control
MIME	Multipurpose Internet Mail Extensions
NFP	Network Forwarding Path
NFPD	NFP Descriptor
NFV	Network Functions Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	NFV Orchestrator
NS	Network Service
NSD	Network Service Descriptor
OSS	Operation Support System
PKG	Package
PM	Performance Management
PNF	Physical Network Function
PNFD	Physical Network Function Descriptor
RAM	Random-Access Memory
REST	Representational State Transfer
RFC	Request For Comments
SAP	Service Access Point
SAPD	Service Access Point Descriptor
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VDU	Virtualisation Deployment Unit
VIM	Virtualised Infrastructure Manager
VL	Virtual Link
VLAN	Virtual Local Area Network
VLD	VL Descriptor
VNF	Virtualised Network Function
VNFC	VNF Component
VNFD	VNF Descriptor
VNFFG	VNF Forwarding Graph
VNFFGD	VNFFG Descriptor
VNFM	VNF Manager
YAML	YAML Ain't Markup Language

4 General Aspects

4.1 Overview

The present document defines the protocol and data model for the following interfaces, in the form of RESTful Application Programming Interface (APIs) specifications:

- NSD Management interface (as produced by the NFVO towards the OSS/BSS)
- NS Lifecycle Management interface (as produced by the NFVO towards the OSS/BSS)
- NS Performance Management interface (as produced by the NFVO towards the OSS/BSS)
- NS Fault Management interface (as produced by the NFVO towards the OSS/BSS)
- VNF Package Management interface (as produced by the NFVO towards the OSS/BSS)

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 013 [3]. In clause 4, general aspects such as URI structure and supported content formats, general procedures and common data types are specified.

In the subsequent clauses, the protocol and data model 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 013 [3].

Even though the various 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, except the "API versions" resource which shall follow the rules specified in clause 4.6.3.

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

where:

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

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

{apiName}	indicates the interface name in an abbreviated form. The {apiName} of each interface is defined in the clause specifying the corresponding interface.
{apiMajorVersion}	indicates the current major version (see clause 4.6.1) of the API and is defined in the clause specifying the corresponding interface.

For HTTP requests and responses that have a body, the content format JSON (see IETF RFC 8259 [20]) 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 [8]) TLS version 1.2 as defined by IETF RFC 5246 [15] 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.1].

All resource URIs of the API shall comply with the URI syntax as defined in IETF RFC 3986 [10]. An implementation that dynamically generates resource URI parts (individual path segments, sequence of path segments that are separated by "/", query parameter values) shall ensure that these parts only use the character set that is allowed by IETF RFC 3986 [10] 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 [10].

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

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

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

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

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

EXAMPLE 1: Objects

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

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

EXAMPLE 2: Unfiltered GET

Request:

```
GET …/container
```

Response:

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

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

EXAMPLE 3: GET with filter

Request:

```
GET …/container?filter=(eq,weight,100)
```

Response:

```
[
    {id:123, weight:100, parts:[{id:1, color:red}, {id:2, color:green}]}
]
```

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

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

4.3.2.2 Specification

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

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

where:

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

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

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

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

The operators listed in Table 4.3.2.2-1 shall be supported.

Table 4.3.2.2-1: Attribute filter operators

Operator with parameters	Meaning
eq,<attrName>,<value>	Attribute **equal** to <value>
neq,<attrName>,<value>	Attribute **not equal** to <value>
in,<attrName>,<value>[,<value>]*	Attribute equal to one of the values in the list ("**in set**" relationship)
nin,<attrName>,<value>[,<value>]*	Attribute not equal to any of the values in the list ("**not in set**" relationship)
gt,<attrName>,<value>	Attribute **greater than** <value>
gte,<attrName>,<value>	Attribute **greater than or equal** to <value>
lt,<attrName>,<value>	Attribute **less than** <value>
lte,<attrName>,<value>	Attribute **less than or equal** to <value>
cont,<attrName>,<value>[,<value>]*	String attribute **contains** (at least) one of the values in the list
ncont,<attrName>,<value>[,<value>]*	String attribute **does not contain** any of the values in the list

Table 4.3.2.2-2: Applicability of the operators to data types

Operator	String	Number	DateTime	Enumeration	Boolean
eq	x	x	-	x	x
neq	x	x	-	x	x
in	x	x	-	x	-
nin	x	x	-	x	-
gt	x	x	x	-	-
gte	x	x	x	-	-
lt	x	x	x	-	-
lte	x	x	x	-	-
cont	x	-	-	-	-
ncont	x	-	-	-	-

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 of type Number, String, Boolean, Enum or DateTime. The content of a <value> entry shall be formatted the same way as the representation of the related attribute in the resource representation. The syntax of DateTime <value> entries shall follow the "date-time" production of IETF RFC 3339 [9]. The syntax of Boolean and Number <value> entries shall follow IETF RFC 8259 [20].

A <value> entry of type String shall be enclosed in single quotes (') if it contains any of the characters ")", "'" or ",", and may be enclosed in single quotes otherwise. Any single quote (') character contained in a <value> entry shall be represented as a sequence of two single quote characters.

The "/" and "~" characters in <attrName> shall be escaped according to the rules defined in section 3 of IETF RFC 6901 [32]. The "," character in <attrName> shall be escaped by replacing it with "~a".

In the resulting <filterExpr>, percent-encoding as defined in IETF RFC 3986 [10] shall be applied to the characters that are not allowed in a URI query part according to Appendix A of IETF RFC 3986 [10], and to the ampersand "&" character.

NOTE: In addition to the statement on percent-encoding above, it is reminded that the percent "%" character is always percent-encoded when used in parts of a URI, according to IETF RFC 3986 [10].

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

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.

The "/" and "~" characters in attribute names in an attribute selector shall be escaped according to the rules defined in section 3 of IETF RFC 6901 [32]. The "," character in attribute names in an attribute selector shall be escaped by replacing it with "~a". Further, percent-encoding as defined in IETF RFC 3986 [10] shall be applied to the characters that are not allowed in a URI query part according to Appendix A of IETF RFC 3986 [10], and to the ampersand "&" character.

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 document for the particular resource.
exclude_default and include=<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 document 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 NFVO implementation. Performing a GET request on such a link shall return a representation that contains the content of the omitted attribute.

EXAMPLE:

```
"_links" : [
    {"vnfs" : {"href" : ".../nslcm/v1/ns_instances/1234/vnfs"}},
    {"virtualLinks" : {"href" : ".../nslcm/v1/ns_instances/1234/virtualLinks"}}
    ]
```

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 [21]	application/json	Content-Types that are acceptable for the response. This header field shall be present if the response is expected to have a non-empty message body.
Content-Type	IETF RFC 7231 [21]	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 [24]	Bearer mF_9.B5f-4.1JqM	The authorization token for the request. Details are specified in clause 4.5.3.
Range	IETF RFC 7233 [23]	1 000-2 000	Requested range of bytes from a file.
Version	IETF RFC 4229 [33]	1.2.0 or 1.2.0-impl:example.com:myNFVO:4	Version of the API requested to use when responding to this request.

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 [21]	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 [21]	http://www.example.com/vnflcm/v1/vnf_instances/123	Used in redirection, or when a new resource has been created. This header field shall be present if the response status code is 201 or 3xx. In the present document this header field is also used if the response status code is 202 and a new resource was created.
WWW-Authenticate	IETF RFC 7235 [24]	Bearer realm="example"	Challenge if the corresponding HTTP request has not provided authorization, or error details if the corresponding HTTP request has provided an invalid authorization token.
Accept-Ranges	IETF RFC 7233 [23]	bytes	Used by the server to signal whether or not it supports ranges for certain resources.
Content-Range	IETF RFC 7233 [23]	bytes 21010-47021/ 47022	Signals the byte range that is contained in the response, and the total length of the file.
Retry-After	IETF RFC 7231 [21]	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.
Version	IETF RFC 4229 [33]	1.2.0 or 1.2.0-impl:example.com:myNFVO:4	Version of the API used in the response.
Link	IETF RFC 8288 [34]	<http://example.com/resources?nextpage_opaque_marker=abc123>; rel="next"	Reference to other resources. Used for paging in the present document, see clause 4.7.2.1.

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 [21], IETF RFC 7232 [22], IETF RFC 7233 [23] and IETF RFC 7235 [24], as well as by IETF RFC 6585 [17]). In addition, the response body should contain a JSON representation of a "ProblemDetails" data structure according to IETF RFC 7807 [27] that provides additional details of the error. In that case, as defined by IETF RFC 7807 [27], 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 [27] is reproduced in Table 4.3.5.3-1. Compared to the general framework defined in IETF RFC 7807 [27], the "status" and "detail" attributes are mandated to be included by the present document, to ensure that the response contains additional textual information about an error. IETF RFC 7807 [27] 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 [27].

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 [10] 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 [27].

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.

NOTE 1: The error handling defined in this clause only applies to REST resources defined in the present document. For the token endpoint defined in IETF RFC 6749 [18] 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.

400 Bad Request: If the request is malformed or syntactically incorrect (e.g. if the request URI contains incorrect query parameters or the payload body contains a syntactically incorrect data structure), 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 response to a GET request which queries a container resource would be so big that the performance of the API producer is adversely affected, and the API producer does not support paging for the affected resource, it 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 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 2: It is by design to represent these application error situations with the same HTTP error response code 400.

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 [19] and IETF RFC 7235 [24]. The ProblemDetails structure may be provided.

NOTE 3: The use of this HTTP error response code described above is applicable to the use of the OAuth 2.0 for the authorization of API requests and notifications, as defined in clauses 4.5.3.3 and 4.5.3.4.

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 [19] and IETF RFC 7235 [24]. 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.

NOTE 4: This response code is not appropriate in case the resource addressed by the URI is a container resource which is designed to contain child resources, but does not contain any child resource at the time the request is received. For a GET request to an existing empty container resource, a typical response contains a 200 OK response code and a payload body with an empty array.

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.

406 Not Acceptable: If the "Accept" 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.

413 Payload Too Large: If the payload body of a request is larger than the amount of data the API producer is willing or able to process, it shall respond with this response code, following the provisions in IETF RFC 7231 [21] for the use of the "Retry-After" HTTP header and for closing the connection. The "ProblemDetails" structure may be omitted.

414 URI Too Long: If the request URI of a request is longer than the API producer is willing or able to process, it shall respond with this response code. This condition can e.g. be caused by passing long queries in the request URI of a GET request. The "ProblemDetails" structure may be omitted.

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

429 Too Many Requests: If the API consumer has sent too many requests in a defined period of time and the API producer is able to detect that condition ("rate limiting"), the API producer shall respond with this response code, following the provisions in IETF RFC 6585 [17] for the use of the "Retry-After" HTTP header. The "ProblemDetails" structure shall be provided and shall include in the "detail" attribute more information about the source of the problem.

NOTE 6: The period of time and allowed number of requests are configured within the API producer by means outside the scope of the present document.

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 [21] for the use of the "Retry-After" HTTP header and for the alternative to refuse the connection. The "ProblemDetails" structure may be omitted.

504 Gateway Timeout: If the API producer encounters a timeout while waiting for a response from an upstream server (i.e. a server that the API producer communicates with when fulfilling a request), it should respond with this response code.

Further error situations are defined for specific REST resources and related HTTP methods in the individual APIs specified in subsequent clauses of the present document.

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 [21] IETF RFC 6750 [19] IETF RFC 7235 [24]	Required information for the request was missing, or 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 [24]	Client is required to include valid credentials in the request. See clause 4.5.3.
403	Forbidden	IETF RFC 7231 [21]	The client is not allowed to perform the request on that resource.
404	Not Found	IETF RFC 7231 [21]	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 [21]	See clause 4.3.5.4.
406	Not Acceptable	IETF RFC 7231 [21]	See clause 4.3.5.4.
409	Conflict	IETF RFC 7231 [21]	Another request is in progress that prohibits the fulfilment of the current request, or the current resource state is inconsistent with the request.

Code	Status text	Reference	Explanation
412	Precondition failed	IETF RFC 7232 [22]	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.
413	Payload Too Large	IETF RFC 7231 [21]	The server is refusing to process a request because the request payload is larger than the server is willing or able to process.
414	URI Too Long	IETF RFC 7231 [21]	The server is refusing to process a request because the request URI is longer than the server is willing or able to process.
416	Range Not Satisfiable	IETF RFC 7233 [23]	This code is returned if the requested byte range in the Range HTTP header is not present in the requested resource.
422	Unprocessable Entity	IETF RFC 4918 [14]	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.
429	Too Many Requests	IETF RFC 6585 [17]	The server is refusing to process a request because the client has sent too many requests in a given period of time (rate limiting).
500	Internal Server Error	IETF RFC 7231 [21]	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 [21]	Server is unable to process the request due to internal overload.
504	Gateway Timeout	IETF RFC 7231 [21]	The server did not receive a timely response from an upstream server it needed to access in order to complete the request.

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 entity 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.2].

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 in clause 4 of IETF RFC 8259 [20].

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 Links data type

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI of another resource referenced from a resource. Shall be an absolute URI (i.e. a URI that contains {apiRoot}).

4.4.1.3a Type: NotificationLink

This type represents a link to a resource in a notification, using an absolute or relative URI. It shall comply with the provisions defined in Table 4.4.1.3a-1.

Table 4.4.1.3a-1: Definition of the NotificationLink data type

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI of a resource referenced from a notification. Should be an absolute URI (i.e. a URI that contains {apiRoot}), however, may be a relative URI (i.e. a URI where the {apiRoot} part is omitted) if the {apiRoot} information is not available.

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 8259 [20]. 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" : 0.05,
    "anArray" : [1,2,3],
    "anObject" : {"organization" : "ETSI", "isg" : "NFV", workingGroup" : "SOL"}
}
```

4.4.1.5 Type: NsInstanceSubscriptionFilter

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

Table 4.4.1.5-1: Definition of the NsInstanceSubscriptionFilter data type

Attribute name	Data type	Cardinality	Description
nsdIds	Identifier	0..N	If present, match NS instances that were created based on a NSD identified by one of the nsdId values listed in this attribute. See note 1.
vnfdIds	Identifier	0..N	If present, match NS instances that contain VNF instances that were created based on a VNFD identified by one of the vnfdId values listed in this attribute. See note 1.
pnfdIds	Identifier	0..N	If present, match NS instances that contain PNFs that are represented by a PNFD identified by one of the pnfdId values listed in this attribute. See note 1.
nsInstanceIds	Identifier	0..N	If present, match NS instances with an instance identifier listed in this attribute. See note 2.
nsInstanceNames	String	0..N	If present, match NS instances with a NS Instance Name listed in this attribute. See note 2.
NOTE 1: The attributes "nsdIds", "vnfdIds" and "pnfdIds" are alternatives to reference to NS instances that are created based on certain NSDs, or contain VNF instances that are based on certain VNFDs, or contain PNFs that are based on certain PNFDs in a filter. They should not be used together in the same filter instance, but one alternative should be chosen. NOTE 2: The attributes "nsInstanceIds" and "nsInstanceNames" are alternatives to reference to particular NS Instances in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

4.4.1.6 Type: ResourceHandle

This type represents the information that allows addressing a virtualised resource that is used by a VNF instance or by an NS instance. Information about the resource is available from the VIM. The ResourceHandle type shall comply with the provisions defined in Table 4.4.1.6-1.

Table 4.4.1.6-1: Definition of the ResourceHandle data type

Attribute name	Data type	Cardinality	Description
vimId	Identifier	0..1	Identifier of the VIM under whose control this resource is placed. This attribute shall be present if VNF-related resource management in direct mode is applicable. It shall also be present for resources that are part of an NS instance such as virtual link resources.
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.
NOTE: 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.			

4.4.1.7 Type: ApiVersionInformation

This type represents API version information. It shall comply with the provisions defined in Table 4.4.1.7-1.

Table 4.4.1.7-1: ApiVersionInformation data type

Attribute name	Data type	Cardinality	Description
uriPrefix	String	1	Specifies the URI prefix for the API, in the following form {apiRoot}/{apiName}/{apiMajorVersion}/
apiVersions	Structure (inlined)	1..N	Version(s) supported for the API signalled by the uriPrefix attribute.
>version	String	1	Identifies a supported version. The value of the version attribute shall be a version identifier as specified in clause 4.6.1.
>isDeprecated	Boolean	0..1	If such information is available, this attribute indicates whether use of the version signaled by the version attribute is deprecated (true) or not (false). See note.
>retirementDate	DateTime	0..1	The date and time after which the API version will no longer be supported. This attribute may be included if the value of the isDeprecated attribute is set to true and shall be absent otherwise.
NOTE: A deprecated version is still supported by the API producer but is recommended not to be used any longer. When a version is no longer supported, it does not appear in the response body.			

4.4.2 Simple data types

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

Table 4.4.2-1: Simple data types

Type name	Description
Identifier	An identifier with the intention of being globally unique. Representation: string of variable length.
IdentifierInNs	An identifier that is unique with respect to a NS. Representation: string of variable length.
IdentifierInNsd	An identifier that is unique within a NS descriptor. Representation: string of variable length.
IdentifierInPnf	An Identifier that is unique within respect to a PNF. 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.
DateTime	Date-time stamp. Representation: String formatted as defined by the date-time production in IETF RFC 3339 [9].
Uri	String formatted according to IETF RFC 3986 [10].
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.
IpAddressPrefix	An IPV4 or IPV6 address range in CIDR format. For IPV4 address range, refer to IETF RFC 4632 [12]. For IPV6 address range, refer to IETF RFC 4291 [11].
Version	A Version. Representation: string of variable length.
String	A string as defined in IETF RFC 8259 [20].
Number	A number as defined in IETF RFC 8259 [20].

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 [18]: *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 [18].

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 [18]. 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. 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 [15], including the optional CertificateRequest from server to client.

HTTP over TLS enables authorization based on TLS certificates as an alternative to a token-based approach.

4.5.2 Flows (informative)

4.5.2.0 General

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

Figure 4.5.2.1-1 illustrates the authorization of API requests that the API consumer sends to the API producer.

NOTE 1: 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.

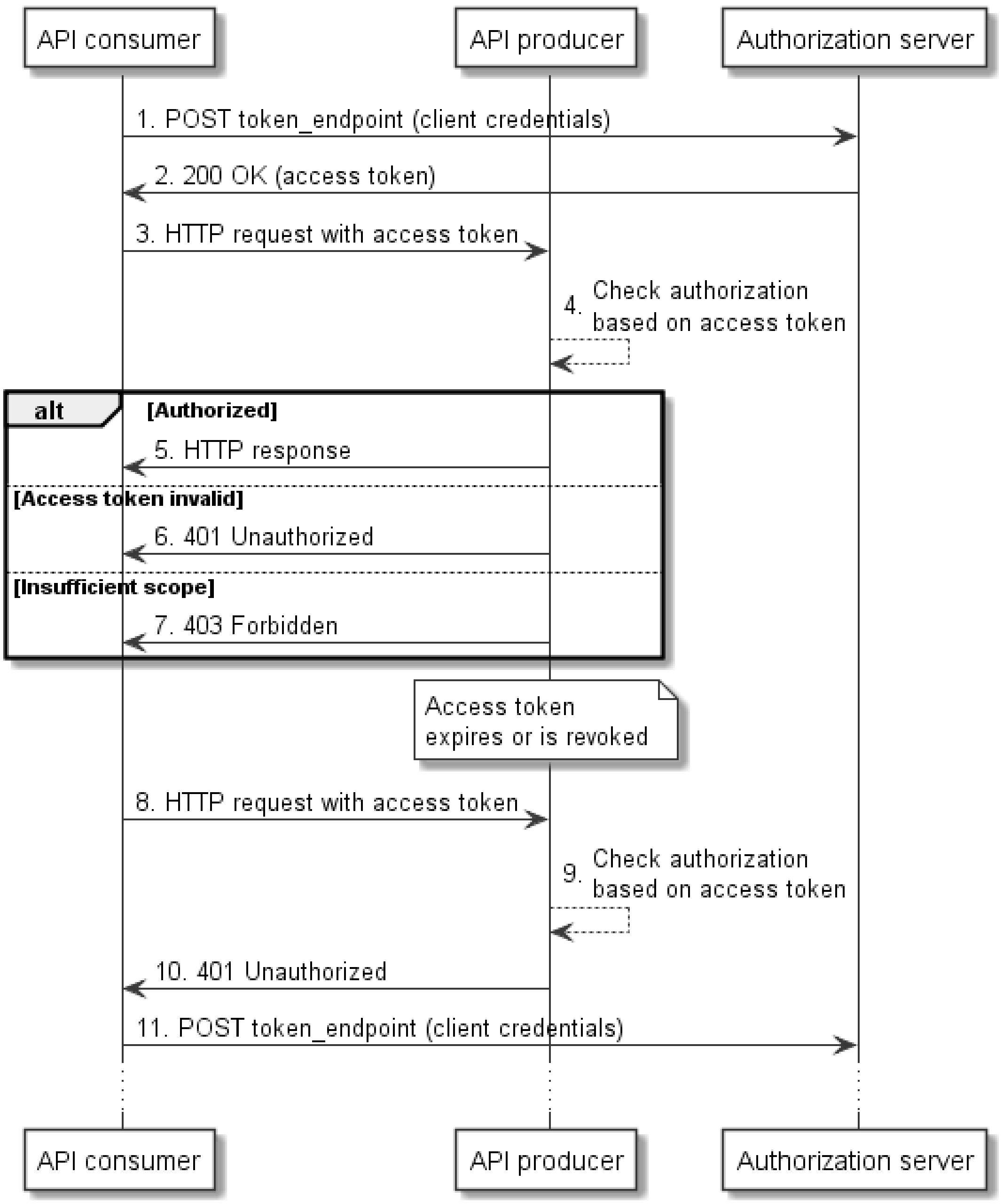
API consumer
API producer
Authorization server
1. POST token_endpoint (client credentials)
2. 200 OK (access token)
3. HTTP request with access token
4. Check authorization based on access token
alt
[Authorized]
5. HTTP response
[Access token invalid]
6. 401 Unauthorized
[Insufficient scope]
7. 403 Forbidden
Access token expires or is revoked
8. HTTP request with access token
9. Check authorization based on access token
10. 401 Unauthorized
11. POST token_endpoint (client credentials)
API consumer
API producer
Authorization server

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

4.5.2.1a 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 [15]. 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 [15].

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.

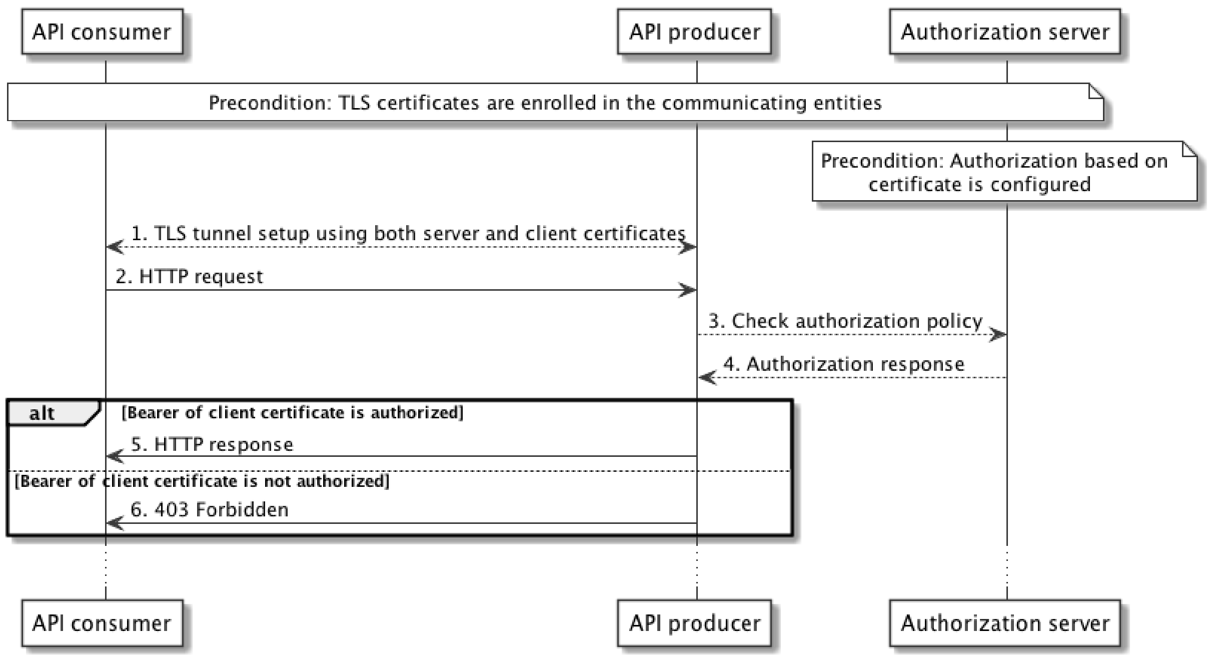
API consumer
API producer
Authorization server
Precondition: TLS certificates are enrolled in the communicating entities
Precondition: Authorization based on certificate is configured
1. TLS tunnel setup using both server and client certificates
2. HTTP request
3. Check authorization policy
4. Authorization response
alt
[Bearer of client certificate is authorized]
5. HTTP response
[Bearer of client certificate is not authorized]
6. 403 Forbidden
API consumer
API producer
Authorization server

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 [15]. 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.

NOTE 3: Authorization based on TLS certificates assumes the existence of a trust relationship between the API producer and the authorization server. The authorization server has no direct communications with the API consumer and thus cannot authenticate it but relies on the API producer to perform this authentication.

4.5.2.2 Authorization of notifications using the HTTP Basic authentication scheme

Figure 4.5.2.2-1 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 [26]). In this flow, no authorization server is needed.

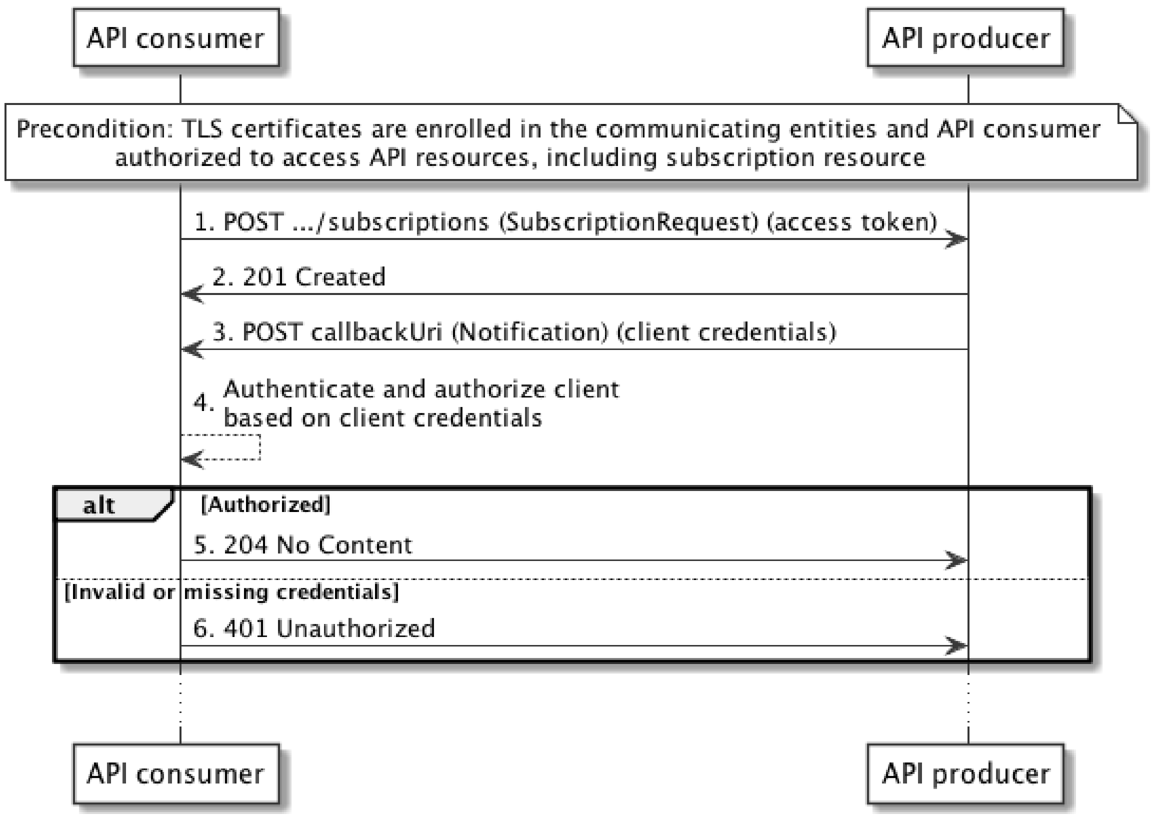
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities and API consumer authorized to access API resources, including subscription resource
1. POST .../subscriptions (SubscriptionRequest) (access token)
2. 201 Created
3. POST callbackUri (Notification) (client credentials)
4. Authenticate and authorize client based on client credentials
alt
[Authorized]
5. 204 No Content
[Invalid or missing credentials]
6. 401 Unauthorized
API consumer
API producer

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 consumer 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 producer returns a "204 No Content" HTTP response to indicate successful delivery of the notification.

6) In case the credentials are invalid, the API producer 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

Figure 4.5.2.3-1 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 1: 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.

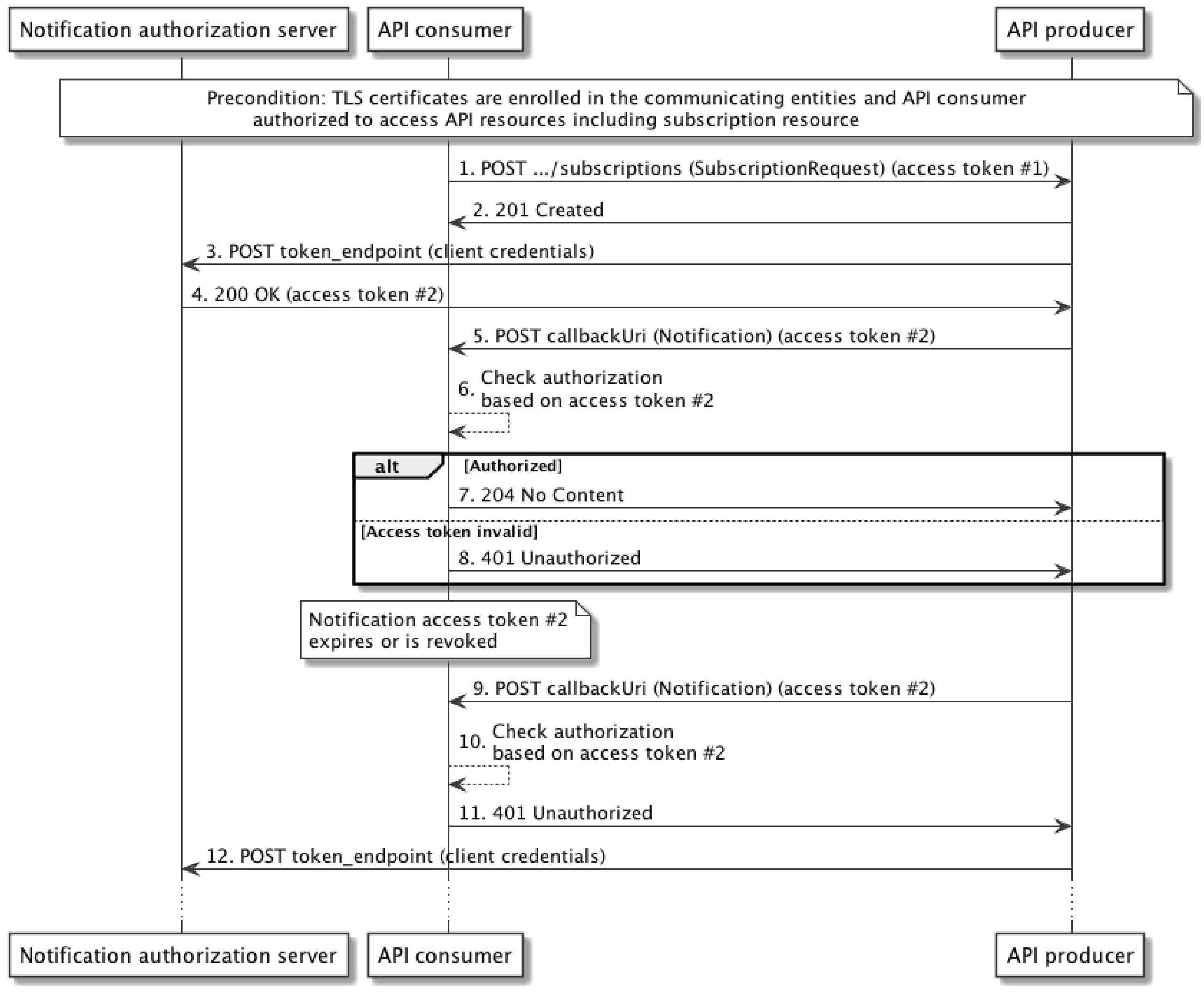
Notification authorization server
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities and API consumer authorized to access API resources including subscription resource
1. POST .../subscriptions (SubscriptionRequest) (access token #1)
2. 201 Created
3. POST token_endpoint (client credentials)
4. 200 OK (access token #2)
5. POST callbackUri (Notification) (access token #2)
6. Check authorization based on access token #2
alt [Authorized]
7. 204 No Content
[Access token invalid]
8. 401 Unauthorized
Notification access token #2 expires or is revoked
9. POST callbackUri (Notification) (access token #2)
10. Check authorization based on access token #2
11. 401 Unauthorized
12. POST token_endpoint (client credentials)

Figure 4.5.2.3-1: Authorization of notifications using OAuth 2.0

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

4.5.2.4 Authorization of notifications using TLS certificates

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.

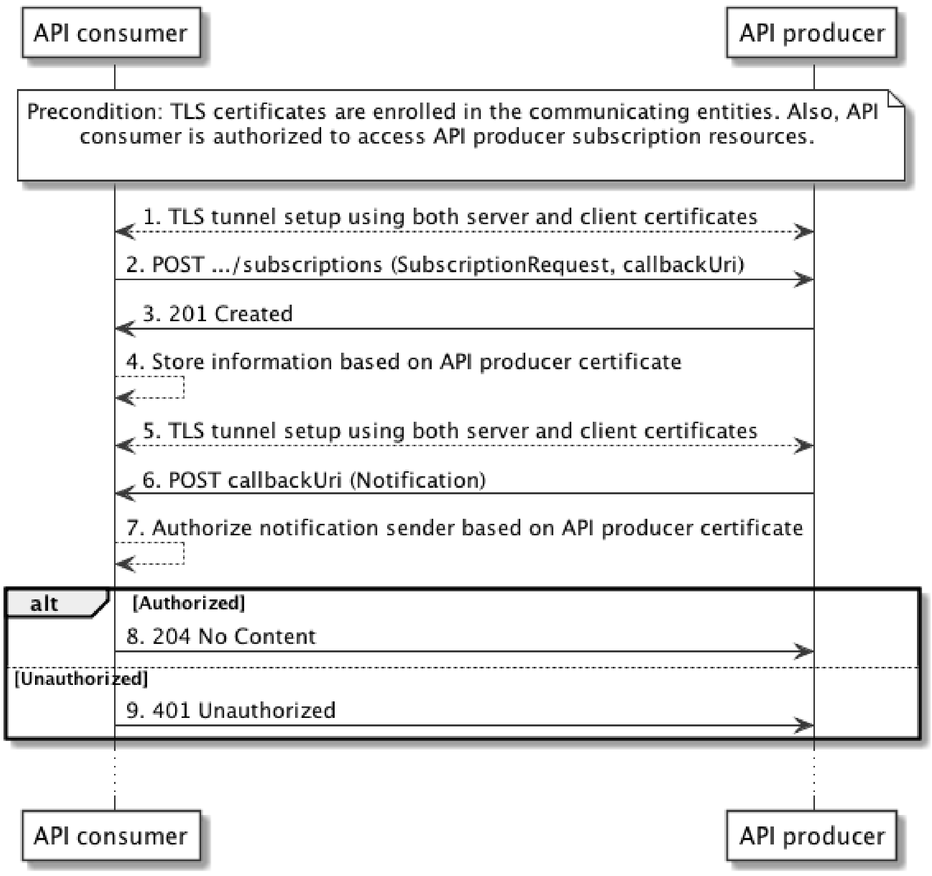
API consumer
API producer
Precondition: TLS certificates are enrolled in the communicating entities. Also, API consumer is authorized to access API producer subscription resources.
1. TLS tunnel setup using both server and client certificates
2. POST .../subscriptions (SubscriptionRequest, callbackUri)
3. 201 Created
4. Store information based on API producer certificate
5. TLS tunnel setup using both server and client certificates
6. POST callbackUri (Notification)
7. Authorize notification sender based on API producer certificate
alt
[Authorized]
8. 204 No Content
[Unauthorized]
9. 401 Unauthorized
API consumer
API producer

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 Os-Ma-nfvo 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 [18]). A *client* shall use the access token request and response according to this grant type, as defined by IETF RFC 6749 [18], to obtain an *access token* for access to the REST resources defined by the present document. 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 [18].

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 [19]. 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 [19].

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 [15], 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 receiving 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.
paramsOauth2ClientCredentials	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.

Attribute name	Data type	Cardinality	Description
>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":

1) 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 [18]. The API consumer should include expiry information with the token response;

2) 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 [19]);

3) 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;

4) 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 [26].

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 NS 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 NFVO or the OSS/BSS. 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, 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, 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, the API consumer and API producer shall use the following procedure, illustrated in Figure 4.5.3.6.1-1, 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, and the API consumer leaves the choice of OAuth or TLS to the API producer.

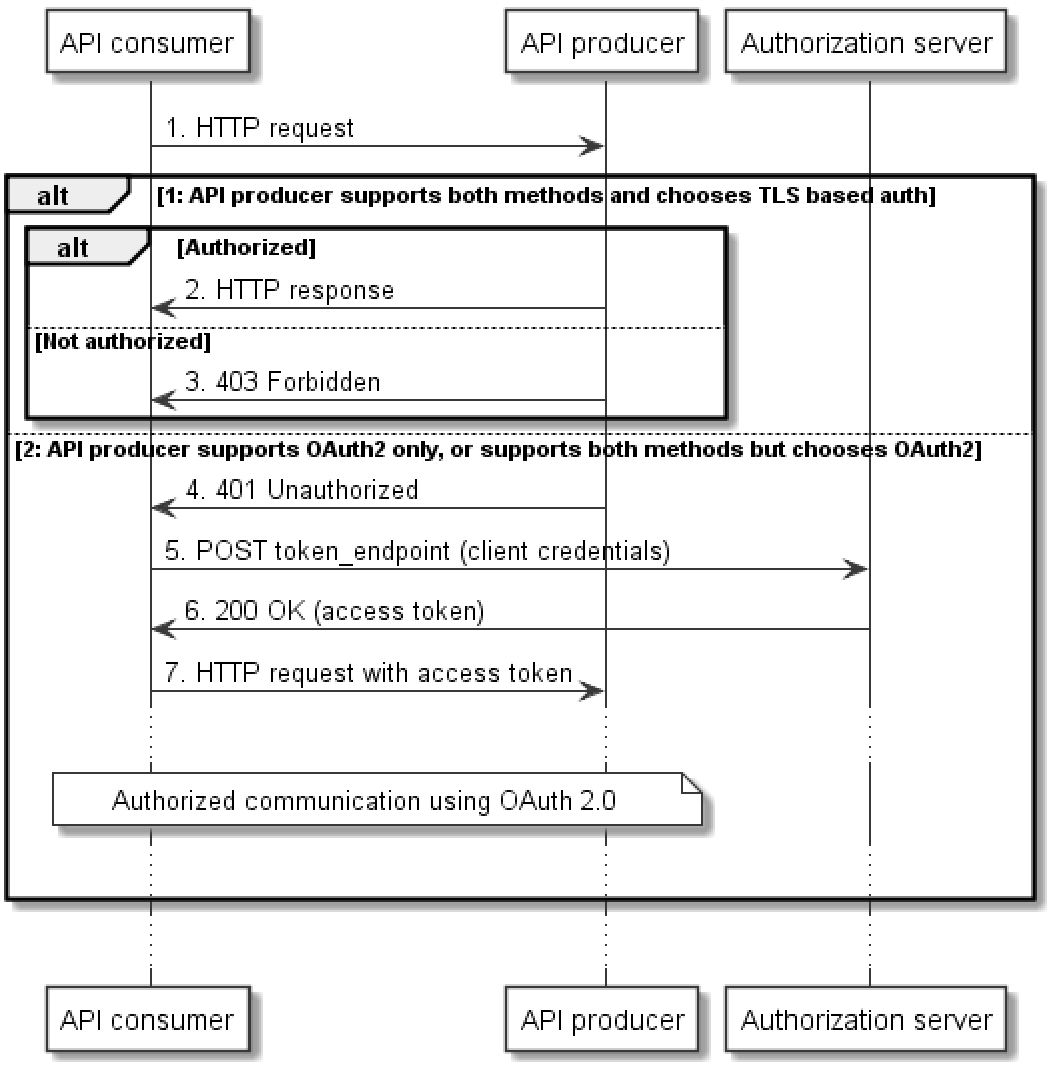
API consumer
API producer
Authorization server
1. HTTP request
alt [1: API producer supports both methods and chooses TLS based auth]
alt [Authorized]
2. HTTP response
[Not authorized]
3. 403 Forbidden
[2: API producer supports OAuth2 only, or supports both methods but chooses OAuth2]
4. 401 Unauthorized
5. POST token_endpoint (client credentials)
6. 200 OK (access token)
7. HTTP request with access token
Authorized communication using OAuth 2.0

Figure 4.5.3.6.1-1: 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 both authorization methods, chooses to use the method 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 both authorization methods, chooses to use the method 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, or chooses to use OAuth 2.0 for authorization, it shall return a 401 Unauthorized response to challenge the API consumer to use OAuth 2.0.

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).

When performing and authorizing an API request and the API consumer does not support the method based on TLS certificates, or supports both methods but decides to use OAuth 2.0, no negotiation takes place, and the method defined in clause 4.5.3.2 shall be used (see also the flow in clause 4.5.2.1).

Table 4.5.3.6.1-1 illustrates the alternatives.

Table 4.5.3.6.1-1: Illustration of the alternatives

Consumer supports	Producer supports	Consumer request	Producer reaction
OAuth2	OAuth2	Consumer sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
OAuth2+TLS	OAuth2	If consumer intends to use OAuth2, it sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
		Otherwise, consumer sends the 1st HTTP request without access token.	Producer sends a 401 challenge to initiate use of OAuth2 (see note 2).
OAuth2	OAuth2+TLS	Consumer sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
OAuth2+TLS	OAuth2+TLS	If consumer intends to use OAuth2, it sends an access token in the 1st HTTP request.	Producer detects that OAuth2 is requested, and sends a success HTTP response if consumer is authorized (see note 1).
		Otherwise, consumer sends the 1st HTTP request without access token.	If producer chooses OAuth2, it sends a 401 challenge (see note 2).
			Otherwise, if producer chooses TLS, it sends a success HTTP response if consumer is authorized (see note 3).
NOTE 1: This flow (OAuth2 method chosen by API consumer) is illustrated in Figure 4.5.2.1-1. NOTE 2: This flow (OAuth2 method chosen by API producer) is illustrated in Figure 4.5.3.6.1-1 as alternative 2. NOTE 3: This flow (TLS method chosen by API producer) is illustrated in Figure 4.5.3.6.1-1 as alternative 1.			

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, 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, 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, the 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 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.

4.6 Version management

4.6.1 Version identifiers and parameters

4.6.1.1 Version identifiers

API version identifiers shall consist of 3 numerical fields, following a MAJOR.MINOR.PATCH pattern and the rules for Semantic Versioning [35] with the additional clarifications defined in clause 4.6.2. The fields in an API version identifier shall be separated by dots ".". The last field may be followed by one or more version parameters.

The MAJOR, MINOR and PATCH fields are defined in [35] for Semantic Versioning.

The {apiMajorVersion} segment of the URIs used by an API shall be set to the character "v" followed by value of the MAJOR field of the API version identifier.

EXAMPLE: ".../nslcm/v1/

The full version identifier (including parameters) is used in ApiVersionInformation (see clause 4.4.1.7) and in version signalling (see clause 4.6.4). Furthermore, it also appears in the corresponding OpenAPI file (see annex C).

4.6.1.2 Version parameters

Version parameters are separated from the version identifier by a dash "-". Version parameters are separated by semicolon ";".

The present document defines the following version parameters:

- impl

The optional "impl" parameter identifies an implementation and a version of this implementation (e.g. implementation delivered by an open source community or a vendor). The OpenAPI specification referenced in annex C is also considered an implementation under this scheme. The "impl" parameter shall have the following structure: "impl:"<vendor>":"<product>":"<impl_version>, where:

- the <vendor> field shall be a string that contains either an IANA Enterprise Number assigned to that vendor, or an Internet domain name owned by that vendor;
- the <product> field shall contain a string identifying the product, chosen by the vendor;
- the <impl_version> field shall contain a number that defines the version of the implementation. Version numbers of subsequent implementations shall be monotonically increasing.

In case of the OpenAPI files provided by ETSI (see annex C), <vendor> shall be set to "etsi.org" and "product" shall be set to "ETSI_NFV_OpenAPI".

4.6.2 Rules for incrementing version identifier fields

4.6.2.1 General

In a REST API, versioning applies to the resources structure (URI structure, URI query parameters, and supported HTTP methods) and the payload body. Different criteria are applied to increment MAJOR, MINOR, and PATCH version fields for changes that affect the URI compared to changes that affect the payload body.

The fields of an API version identifier are incremented from a previous version to the current version according to the following rules:

- 1st field (MAJOR): This field is always incremented when one or more changes made to the resources structure defined in the present document break backward compatibility. This field is also incremented if one or more changes to at least one payload body defined in the present document break backward compatibility, unless that change is correcting an error.

NOTE 1: A change that corrects an error that would lead the API producer to always send an error response if a certain valid condition is met is not considered a non-backward compatible change, irrespective of the type of change. Indeed compatibility between a new version and a previous version can only be assessed for a feature that is properly supported in the previous version.

NOTE 2: The 1st field (MAJOR) is kept equal to "1" in version 2.5.1 of the present document, recognizing that existing and emerging commercial implementations could be adversely affected by a change to the MAJOR version at this time.

- 2nd field (MINOR): This field is incremented if one or more technical changes (at least one of which is not an error correction) are made to the API specification in the present document API but none of them (apart from error corrections to the payload body) breaks backward compatibility. It is reset to zero if the MAJOR version identifier is changed.
- 3rd field (PATCH): This field is incremented if one or more error corrections that are visible in communication between API producer and API consumer are made on the API specification in the present document but none of them (apart from error corrections to the payload body) breaks backward compatibility. It is reset to zero if the MINOR version identifier is changed.

NOTE 3: All the aforementioned types of changes affect the corresponding OpenAPI specification (see annex C).

4.6.2.2 Examples of backward and non-backward compatible changes

Examples of backward compatible changes include:

- Adding a new resource
- Adding a new URI
- Supporting a new HTTP method for an existing resource
- Adding new optional URI query parameters
- Adding new optional attributes to a resource representation in a request
- Adding new attributes to a resource representation in a response or to a notification message
- Responding with a new status code of an error class
- Certain cardinality changes (see note 1)

NOTE 1: Whether attribute cardinality changes are backward compatible depends on the type of change. An example of a backward-compatible cardinality change include making an attribute in a response required (e.g. changing cardinality from 0..1 to 1).

Examples of non-backward compatible changes to the resources structure include:

- Removing a resource / URI
- Removing support for an HTTP method
- Changing a resource URI
- Adding new mandatory URI query parameters

Examples of non-backward compatible changes to the payload body include:

- Renaming an attribute in a resource representation
- Adding new mandatory attributes to a resource representation in a request
- Changing the data type of an attribute
- Certain cardinality changes (see note 2)

NOTE 2: Whether attribute cardinality changes are backward compatible depends on the type of change. Examples of non-backward compatible cardinality changes include decreasing the upper bound of a cardinality range for attributes sent by the client, changing the meaning of the default behaviour associated to the absence of an attribute of cardinality 0..N, etc.

4.6.3 Version information retrieval

4.6.3.1 General

The API producer shall support the following dedicated URIs to enable API consumers to retrieve information about API versions supported by an API producer:

```
1. {apiRoot}/{apiName}/api_versions
2. {apiRoot}/{apiName}/{apiMajorVersion}/api_versions
```

To obtain information about the supported API versions, the API consumer shall send a GET request to a URI of one of above forms. The information contained in the GET response depends on the form of URI used in the GET request, as follows:

- If the first form is used, the GET response shall provide the list of supported versions for the API corresponding to the apiName indicated in the GET Request URI.
- If the second form is used, the GET response shall provide the list of supported versions for the API corresponding to the apiName and the apiVersion indicated in the GET Request URI.
- If the API producer receives a GET request:
    - In case of success, the API producer shall return in the body of a 200 OK response a value of the ApiVersionInformation data type specified in clause 4.4.1.7.
    - In case URI query parameters are provided, the API producer shall return a "400 Bad request" response as defined in clause 4.3.5.4.
    - In other cases of failure, the API producer shall return appropriate error information as defined in clauses 4.3.5.4 and 4.3.5.5.

4.6.3.2 Resource structure and methods

Table 4.6.3.2-1 lists the individual resources defined for supporting API version information retrieval, and the applicable HTTP method. The VNFM shall support responding to GET requests on the resources in Table 4.6.3.2-1.

Table 4.6.3.2-1: Resources and methods overview for API version information retrieval

Resource name	Resource URI	HTTP Method	Meaning
API versions	/{apiName}/api_versions	GET	Version information associated to an API
API versions	/{apiName}/{apiMajorVersion}/api_versions	GET	Version information associated to a major version of an API

Figure 4.6.3.2-1 shows the "API versions" resources in the overall resource URI structure defined for all APIs.

The "API versions" resources, as defined in the present clause, are part of the overall resource URI structure of each API defined in the present document.

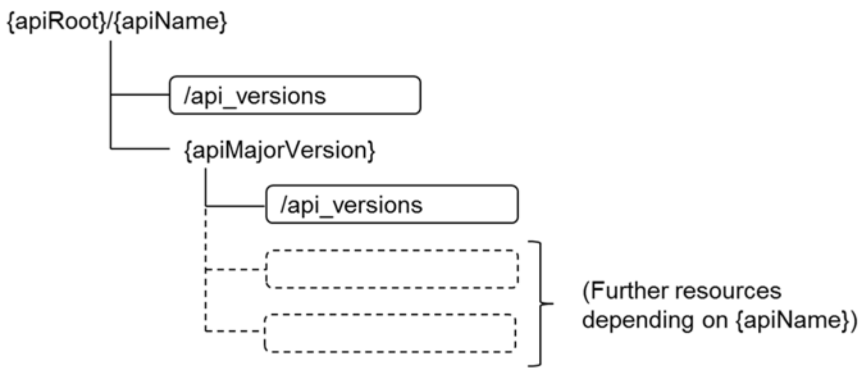

Figure 4.6.3.2-1: "API versions" resources

4.6.3.3 Resource: API versions

4.6.3.3.1 Description

There are two "API versions" resources defined for each API. The client can use these resources to obtain API version information.

4.6.3.3.2 Resource definition

The resource URI of each of the two "API versions" resources shall be of one of the following forms.

1. `{apiRoot}/{apiName}/api_versions`
2. `{apiRoot}/{apiName}/{apiMajorVersion}/api_versions`

These resources shall support the resource URI variables defined in Table 4.6.3.3.2-1.

Table 4.6.3.3.2-1: Resource URI variables for these resources

Name	Definition
apiRoot	See clause 4.2
apiName	See clause 4.2
apiMajorVersion	See clause 4.2

4.6.3.3.3 Resource methods

4.6.3.3.3.1 POST

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.2 GET

The GET method reads API version information. This method shall follow the provisions specified in Table 4.6.3.3.3.2-1 for request and response data structures, and response codes. URI query parameters are not supported.

Table 4.6.3.3.3.2-1: 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**
	ApiVersionInformation	1	200 OK	API version information was read successfully. The response body shall contain API version information, as defined in clause 4.4.1.7.
	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.

4.6.3.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.3.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the VNFM shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

4.6.4 Version signalling

The API consumer shall include the "Version" HTTP header (see IETF RFC 4229 [33]) in each HTTP request. The "Version" header shall contain the three version identifier fields (MAJOR.MINOR.PATCH) indicating the API version the API consumer intends to use. The "impl" version parameter may be provided, indicating the version of the API producer implementation that the API consumer intends to use.

The API producer shall support receiving and interpreting the "Version" HTTP header. The API producer shall include in the response the "Version" HTTP header signalling the used API version, including the "impl" version parameter if available. If the "impl" version parameter has been omitted in the request, the API producer shall use the combination of MAJOR, MINOR and PATCH as requested and the highest supported value for the "impl_version" field of the "impl" version parameter for that combination, if available.

NOTE: In case multiple versions and/or implementation versions are supported by an API producer, this allows the API consumer to request a particular version.

API consumers conforming to versions of the present document previous to version 2.5.1 omit this header. If the API producer receives a request without this header:

- If it supports the previous version 2.4.1 of the present document, it shall behave as defined in that document, and should indicate this by using MAJOR=1 and MINOR=1 and PATCH=0 in the "Version" HTTP header in the response.

- If it does not support any of the previous versions, it shall respond with a 400 Bad Request response and shall include in the response payload body a ProblemDetails structure providing more information on the cause of the error in the "detail" attribute.

If the API version signalled in the "Version" request header is not supported by the API producer, the API producer shall respond with a "406 Not Acceptable" error and shall include in the response payload body a ProblemDetails structure providing more information on the cause of the error in the "detail" attribute.

4.7 Handling of large query results

4.7.1 Description

If the response to a query to a container resource (i.e. a resource that contains child resources whose representations will be returned when responding to a GET request) will become so large that the response will adversely affect the performance of the server, the server either rejects the request with a 400 Bad Request response, or the server provides a paged response, i.e. it returns only a subset of the query result in the response, and also provides information how to obtain the remainder of the query result.

When returning a paged response, depending on the underlying storage organization, it might be problematic for the server to determine the actual size of the result; however, it is usually possible to determine whether there will be additional results returned when knowing, for the last entry in the returned page, the position in the overall query result or some other property that has ordering semantics. For example, the time of creation of a resource has such an ordering property. When using such an (implementation-specific) property, the API producer can correctly handle deletions of child resources that happen between sending the first page of the query result, and sending the next page. It cannot be guaranteed that child resources inserted between returning subsequent pages can be considered in the query result, however, it shall be guaranteed that this does not lead to skipping of entries that have existed prior to insertion.

At minimum, a paged response needs to contain information telling the API consumer that the response is paged, and how to obtain the next page of information. For that purpose, a link to obtain the next page is returned in an HTTP header, containing a parameter that is opaque to the API consumer, but that allows the API producer to determine the start of the next page.

NOTE: In the present document, this functionality is designed for overload protection only. Additional functionality, such as configuring the page size by the API consumer, determining the size of the overall query result or the number of pages, and determining the previous page, is left outside the scope of the present document.

4.7.2 Specification

4.7.2.1 Alternatives

For each container resource (i.e. a resource that contains child resources whose representations will be returned when responding to a GET request), the API producer shall support one of the following two behaviours specified below to handle the case that a response to a query (GET request) will become so large that the response will adversely affect performance:

1) Return an error response, as defined in clause 4.7.2.2.

2) Return the result in a paged manner, as defined in clause 4.7.2.3.

4.7.2.2 Error response

In this alternative, the server shall reject the request with a 400 Bad Request response, shall include the "ProblemDetails" payload body, and shall provide in the "detail" attribute more information about the error.

This error code indicates to the API consumer that with the given attribute-based filtering query (or absence thereof), the response would have been so big that performance is adversely affected. The client can obtain a query result by specifying a (more restrictive) attribute-based filtering query (see clause 4.3.2).

4.7.2.3 Paged response

In this alternative, the API producer shall provide a response that contains a first page (subset) of the results to the query, and shall include a LINK HTTP header (see IETF RFC 8288 [34]) with the "rel" attribute set to "next", which communicates a URI that allows to obtain the next page of results to the original query. The API consumer can send a GET request to the URI communicated in the LINK header to obtain the next page of results. The response which returns that next page shall contain the LINK header to point to the next page, as specified above, unless there are no further pages available in which case the LINK header shall be omitted.

To allow the API producer to determine the start of the next page, the LINK header shall contain the URI query parameter "nextpage_opaque_marker" whose value is chosen by the API producer. This parameter has no meaning for the API consumer, but is echoed back by the API consumer to the API producer when requesting the next page. The URI in the link header may include further parameters, such as those passed in the original request.

The size of each page may be chosen by the API provider, and may vary from page to page. The maximum page size is determined by means outside the scope of the present document.

The response need not contain entries that correspond to child resources which were created after the original query was issued.

5 NSD Management interface

5.1 Description

This interface allows the OSS/BSS to invoke management operations of NSDs towards the NFVO and to subscribe to notifications related to NSD management changes.

The operations provided through this interface are as follows:

- Create NSD Info
- Upload NSD
- Fetch NSD
- Update NSD Info
- Delete NSD
- Query NSD Info
- Create PNFD Info
- Upload PNFD
- Fetch PNFD
- Update PNFD Info
- Delete PNFD
- Query PNFD Info
- Subscribe
- Terminate Subscription
- Query Subscription Information
- Notify

This interface also enables to invoke error handling procedures (i.e., Retry, Rollback, Continue, Cancel, and Fail) on the actual NS lifecycle management operation occurrences, and API version retrieval.

The state changes of a NSD are illustrated in clause B.2.

5.1a API version

For the NSD management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 1, and the PATCH version number shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.4.1 of the present document.

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 "nsd" shall be used to represent {apiName}. 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 NSD management interface.

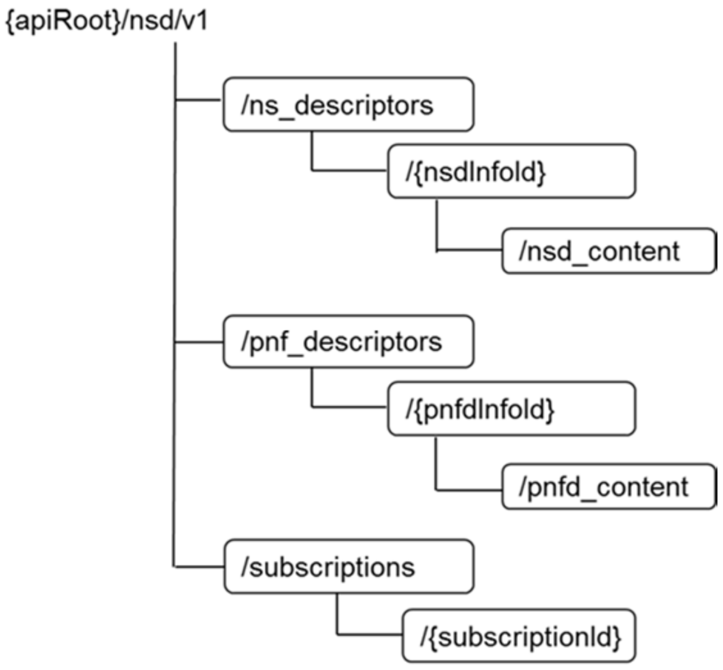

Figure 5.2-1: Resource URI structure of NSD Management Interface

Table 5.2-1 lists the individual resources defined, and the applicable HTTP methods. The NFVO 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. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

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

Resource name	Resource URI	HTTP Method	Cat	Meaning
NS Descriptors	/ns_descriptors	GET	M	Query information about multiple NS descriptor resources.
		POST	M	Create a new NS descriptor resource.
Individual NS Descriptor	/ns_ descriptors/{nsdInfoId}	GET	M	Read information about an individual NS descriptor resource.
		PATCH	M	Modify the operational state and/or the user defined data of an individual NS descriptor resource.
		DELETE	M	Delete an individual NS descriptor resource.
NSD Content	/ns_descriptors/{nsdInfoId}/nsd_content	GET	M	Fetch the content of a NSD.
		PUT	M	Upload the content of a NSD.

Resource name	Resource URI	HTTP Method	Cat	Meaning
PNF Descriptors	/pnf_descriptors	GET	M	Query information about multiple PNF descriptor resources.
		POST	M	Create a new PNF descriptor resource.
Individual PNF Descriptor	/pnf_descriptors/{pnfdInfoId}	GET	M	Read an individual PNFD resource.
		PATCH	M	Modify the user defined data of an individual PNF descriptor resource.
		DELETE	M	Delete an individual PNF descriptor resource.
PNFD Content	/pnf_descriptors/{pnfdInfoId}/pnfd_content	GET	M	Fetch the content of a PNFD.
		PUT	M	Upload the content of a PNFD.
Subscriptions	/subscriptions	POST	M	Subscribe to NSD and PNFD 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	See note	Notify about NSD and PNFD changes. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the OSS/BSS. If the OSS/BSS supports invoking the POST method on the "Subscriptions" resource towards the NFVO, 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 an individual NS descriptor resource

This clause describes the procedure for creating an individual NS descriptor resource.

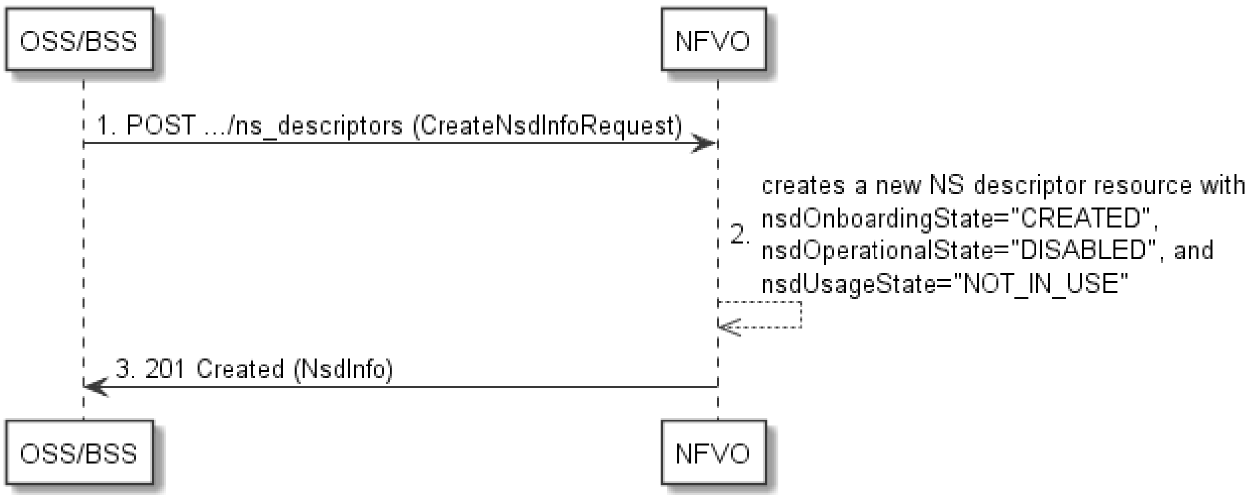
OSS/BSS
NFVO
1. POST .../ns_descriptors (CreateNsdInfoRequest)
2. creates a new NS descriptor resource with nsdOnboardingState="CREATED", nsdOperationalState="DISABLED", and nsdUsageState="NOT_IN_USE"
3. 201 Created (NsdInfo)
OSS/BSS
NFVO

Figure 5.3.1-1: Flow of the creation of an individual NS descriptor resource

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

1) The OSS/BSS sends a POST request to the "ns_descriptors" resource including in the payload body a data structure of type "CreateNsdInfoRequest".

2) The NFVO creates a new NS descriptor resource with nsdOnboardingState="CREATED", nsdOperationalState="DISABLED" and nsdUsageState="NOT_IN_USE".

3) The NFVO returns a 201 Created response containing a representation of the individual NS descriptor resource just created by the NFVO.

Postcondition: Upon successful completion, the individual NS descriptor resource has been created with nsdOnboardingState="CREATED", nsdOperationalState="DISABLED", and nsdUsageState="NOT_IN_USE".

5.3.2 Flow of the uploading of NSD content

This clause describes the procedure for the uploading of NSD content.

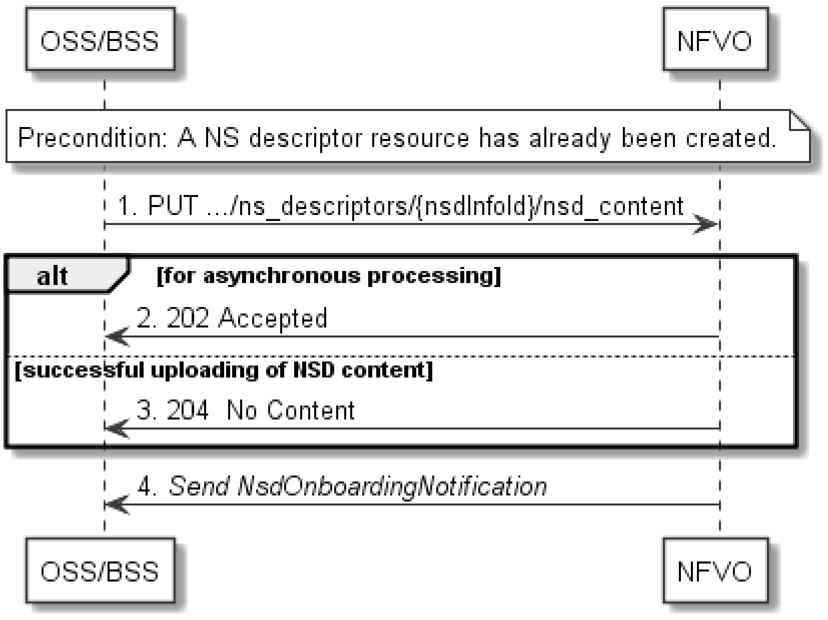
OSS/BSS
NFVO
Precondition: A NS descriptor resource has already been created.
1. PUT .../ns_descriptors/{nsdInfoId}/nsd_content
alt
[for asynchronous processing]
2. 202 Accepted
[successful uploading of NSD content]
3. 204 No Content
4. Send NsdOnboardingNotification
OSS/BSS
NFVO

Figure 5.3.2-1: Flow of the uploading of NSD content

NOTE: Due to possible race conditions, the 204 response and the NsdOnBoardingNotification can arrive in any order at the OSS/BSS.

Precondition: A NS descriptor resource has already been created.

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

1) The OSS/BSS sends a PUT request to a "NSD Content" resource using a "Content-Type" HTTP header as defined in clause 5.4.4.3.3 of the present document.

2) For asynchronous processing, the NFVO returns "202 Accepted".

3) Otherwise, the NFVO returns a "204 No Content" response to the OSS/BSS with an empty payload body for successful uploading of the NSD content.

The NFVO sends an NsdOnboardingNotification to the OSS/BSS.

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

5.3.3 Flow of the fetching of NSD content

This clause describes the procedure for fetching the content of an onboarded NSD.

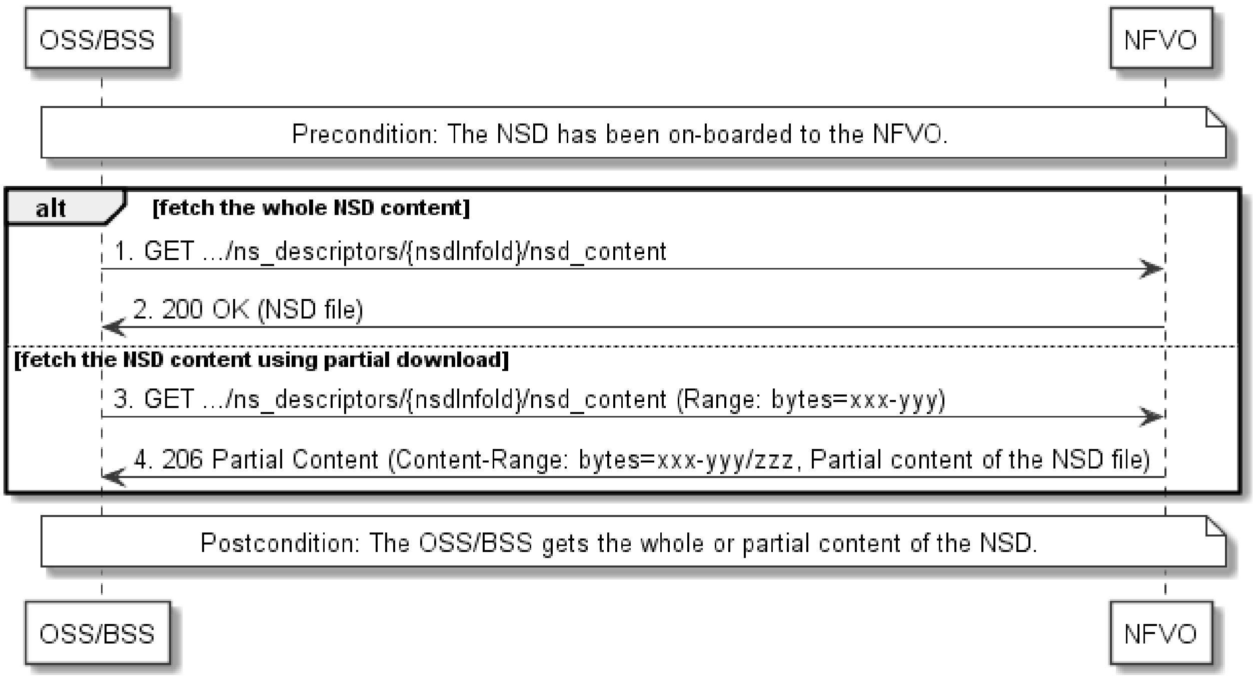
OSS/BSS
NFVO
Precondition: The NSD has been on-boarded to the NFVO.
alt
[fetch the whole NSD content]
1. GET .../ns_descriptors/{nsdInfoId}/nsd_content
2. 200 OK (NSD file)
[fetch the NSD content using partial download]
3. GET .../ns_descriptors/{nsdInfoId}/nsd_content (Range: bytes=xxx-yyy)
4. 206 Partial Content (Content-Range: bytes=xxx-yyy/zzz, Partial content of the NSD file)
Postcondition: The OSS/BSS gets the whole or partial content of the NSD.
OSS/BSS
NFVO

Figure 5.3.3-1: Flow of the fetching of NSD content

Precondition: The NSD is on-boarded to the NFVO.

Fetching an on-boarded NSD, as illustrated in Figure 5.3.3-1, consists of the following steps:

1) If fetching the whole NSD content, the OSS/BSS sends a GET request to the "NSD content" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the NSD file in the payload body.

3) If fetching the NSD content using partial download, the OSS/BSS sends a GET request to the "NSD content" resource, and includes a "Range" HTTP header indicating the partition of the NSD content that needs to be transferred.

4) The NFVO returns a "206 Partial Content" response with a payload body containing the partial content of the NSD, and a "Content-Range" HTTP header indicating the byte range enclosed in the payload and the complete length of the NSD.

Postcondition: Upon successful completion, the OSS/BSS gets the whole or partial content of the NSD.

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

5.3.4 Flow of the update of an individual NS descriptor resource

This clause describes the procedure for the update of an NS descriptor resource. The Update NSD Info operation allows for the modification of the operational state and/or user defined data of an individual NS descriptor resource.

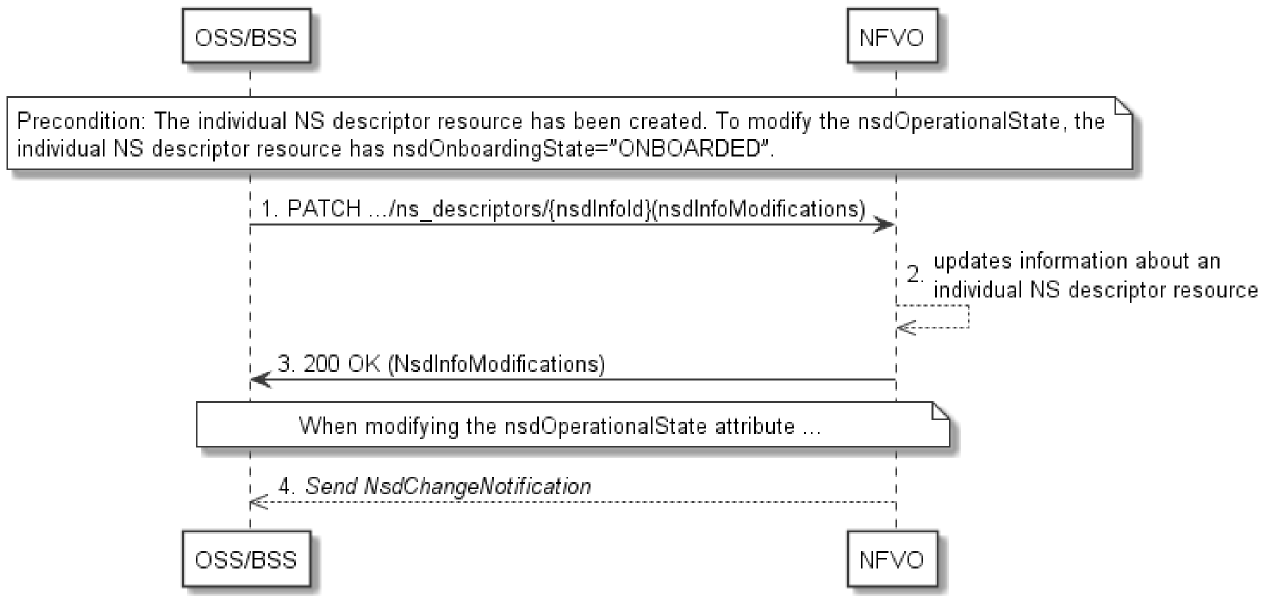
OSS/BSS
NFVO
Precondition: The individual NS descriptor resource has been created. To modify the nsdOperationalState, the individual NS descriptor resource has nsdOnboardingState="ONBOARDED".
1. PATCH .../ns_descriptors/{nsdInfoId}(nsdInfoModifications)
2. updates information about an individual NS descriptor resource
3. 200 OK (NsdInfoModifications)
When modifying the nsdOperationalState attribute ...
4. Send NsdChangeNotification
OSS/BSS
NFVO

Figure 5.3.4-1: Flow of the update of an individual NS descriptor resource

NOTE: Due to possible race conditions, the 200 response and the NsdChangeNotification can arrive in any order at the OSS/BSS.

Precondition: The individual NS descriptor resource has been created. To modify the nsdOperationalState from "ENABLED" to "DISABLED" or vice-versa, the individual NS descriptor resource has nsdOnboardingState="ONBOARDED".

The procedure consists of the following steps as illustrated in Figure 5.3.4-1:

1) The OSS/BSS sends a PATCH request to the "Individual NS descriptor" resource.

2) The NFVO modifies the information associated with the individual NS descriptor resource.

3) The NFVO returns a "200 OK" response including the data structure of type "nsdInfoModifications" in the payloadbody.

4) When modifying the nsdOperationalState attribute, the NFVO sends to the OSS/BSS a NsdChangeNotification to indicate the state change of the individual NS descriptor resource.

Postcondition: Upon successful completion, the information about an individual NS descriptor resource has been updated.

5.3.5 Flow of the deletion of an individual NS descriptor resource

This clause describes the procedure for the deletion of an individual NS descriptor resource.

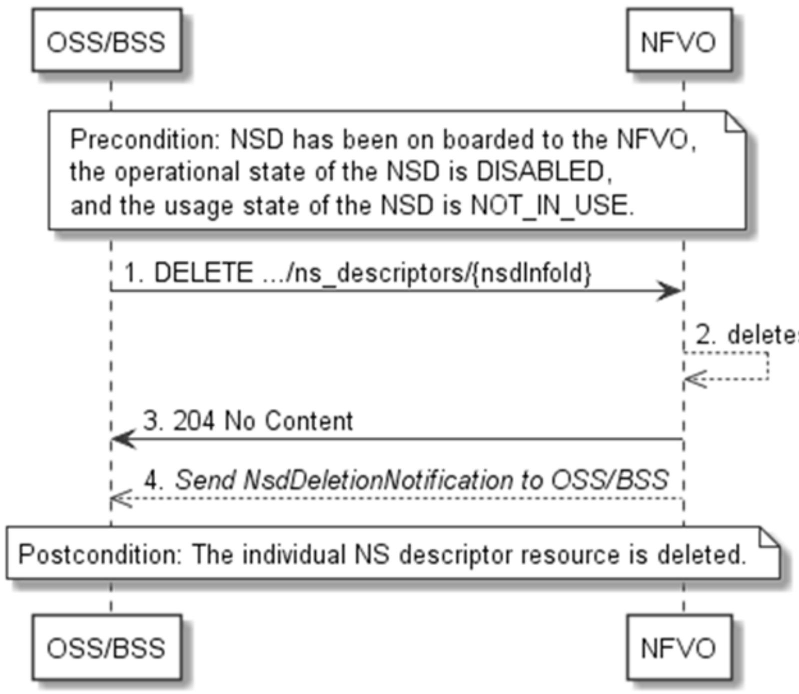
OSS/BSS
NFVO
Precondition: NSD has been on boarded to the NFVO, the operational state of the NSD is DISABLED, and the usage state of the NSD is NOT_IN_USE.
1. DELETE .../ns_descriptors/{nsdInfoId}
2. deletes an individual NS descriptor resource
3. 204 No Content
4. Send NsdDeletionNotification to OSS/BSS
Postcondition: The individual NS descriptor resource is deleted.
OSS/BSS
NFVO

Figure 5.3.5-1: Flow of the deletion of an individual NS descriptor resource

NOTE: Due to possible race conditions, the 204 response and the NsdDeletionNotification can arrive in any order at the OSS/BSS.

Precondition: NSD has been on boarded to the NFVO, the operational state of the NSD is equal to "DISABLED", and the usage state of the NSD is equal to "NOT_IN_USE".

The procedure consists of the following steps as illustrated in Figure 5.3.5-1:

1) The OSS/BSS sends a DELETE request to an "Individual NS descriptor" resource.

2) The NFVO deletes the individual NS descriptor resource.

3) The NFVO returns a "204 No Content" response to the OSS/BSS with an empty payload body.

4) The NFVO sends to the OSS/BSS a NsdDeletionNotification to indicate the deletion of the individual NS descriptor resource.

5.3.6 Flow of the querying/reading of NS descriptor resources

This clause describes the procedure for querying information about multiple NS descriptor resources and reading information about an individual NS descriptor resource.

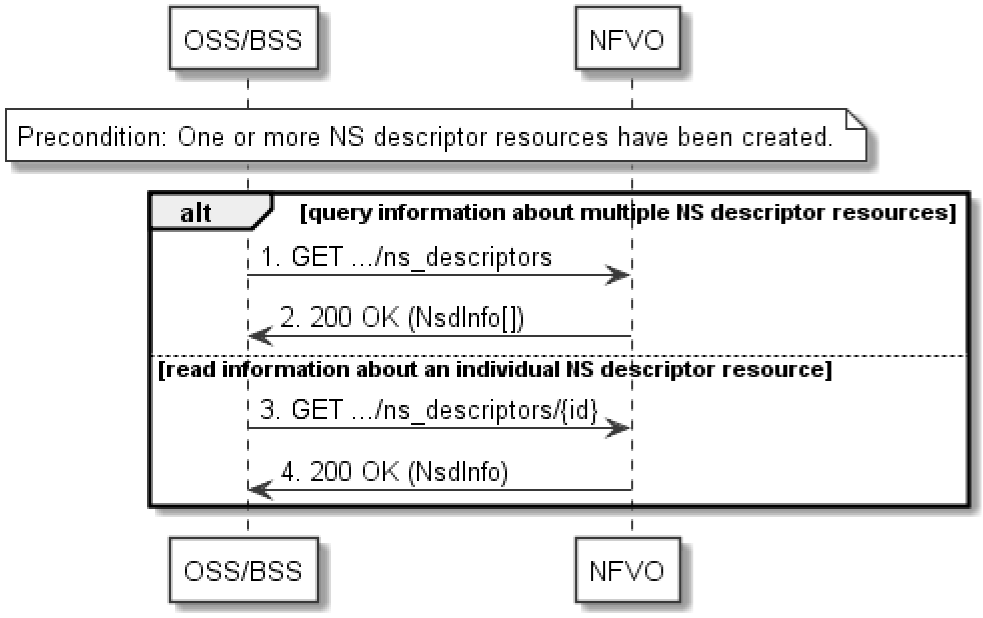
OSS/BSS
NFVO
Precondition: One or more NS descriptor resources have been created.
alt
[query information about multiple NS descriptor resources]
1. GET .../ns_descriptors
2. 200 OK (NsdInfo[])
[read information about an individual NS descriptor resource]
3. GET .../ns_descriptors/{id}
4. 200 OK (NsdInfo)
OSS/BSS
NFVO

Figure 5.3.6-1: Flow of the querying/reading of NS descriptor resources

The procedure consists of the following steps as illustrated in Figure 5.3.6-1.

Precondition: One or more NS descriptor resources have been created:

1) If the OSS/BSS intends to query information about multiple NS descriptor resources, it sends a GET request to the ns_descriptors resource.

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

3) If the OSS/BSS intends to read information about an individual NS descriptor resource, the OSS/BSS sends a GET request to the "Individual NS descriptor" resource, addressed by the appropriate NsdInfo identifier in its resource URI.

4) The NFVO returns a "200 OK" response, and includes in the payload body a data structure of type "NsdInfo".

Postcondition: Upon successful completion, the OSS/BSS gets the information of multiple (i.e. zero or more) NS descriptor resources or an individual NS descriptor resource.

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

5.3.7 Flow of the creation of an individual PNF descriptor resource

This clause describes the procedure for creating an individual PNF descriptor resource.

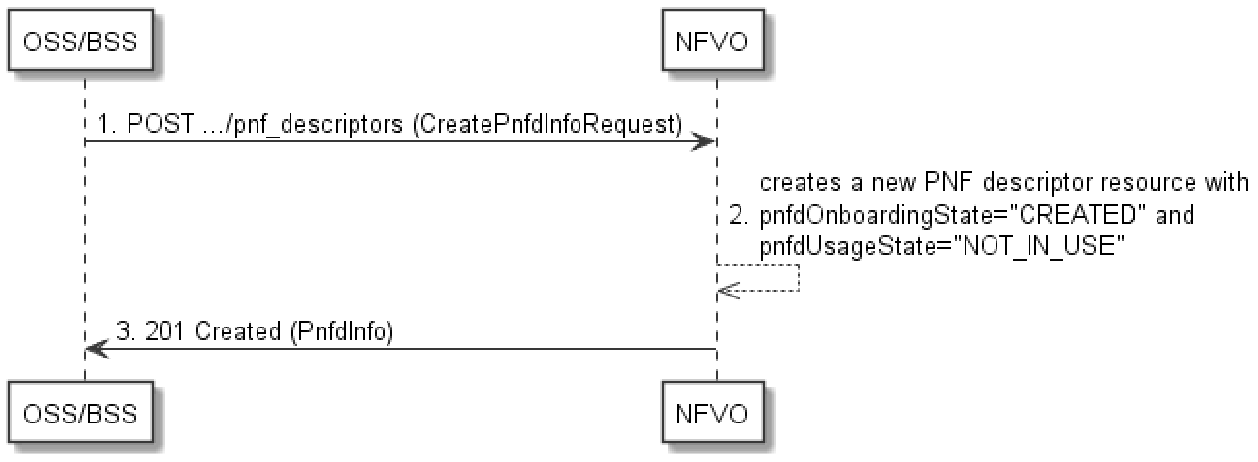
OSS/BSS
NFVO
1. POST .../pnf_descriptors (CreatePnfdInfoRequest)
2. creates a new PNF descriptor resource with pnfdOnboardingState="CREATED" and pnfdUsageState="NOT_IN_USE"
3. 201 Created (PnfdInfo)
OSS/BSS
NFVO

Figure 5.3.7-1: Flow of the creation of an individual PNF descriptor resource

The procedure consists of the following steps as illustrated in Figure 5.3.7-1:

1) The OSS/BSS sends a POST request to the "pnf_descriptors" resource including in the payload body a data structure of type "CreatePnfdInfoRequest".

2) The NFVO creates a new PNF descriptor resource with pnfdOnboardingState="CREATED" and pnfdUsageState="NOT_IN_USE".

3) The NFVO returns a 201 Created response containing a representation of the individual PNF descriptor resource just created by the NFVO.

Postcondition: Upon successful completion, the individual PNF descriptor resource has been created with pnfdOnboardingState="CREATED" and pnfdUsageState="NOT_IN_USE".

5.3.8 Flow of the uploading of PNFD content

This clause describes the procedure for the uploading of PNFD content.

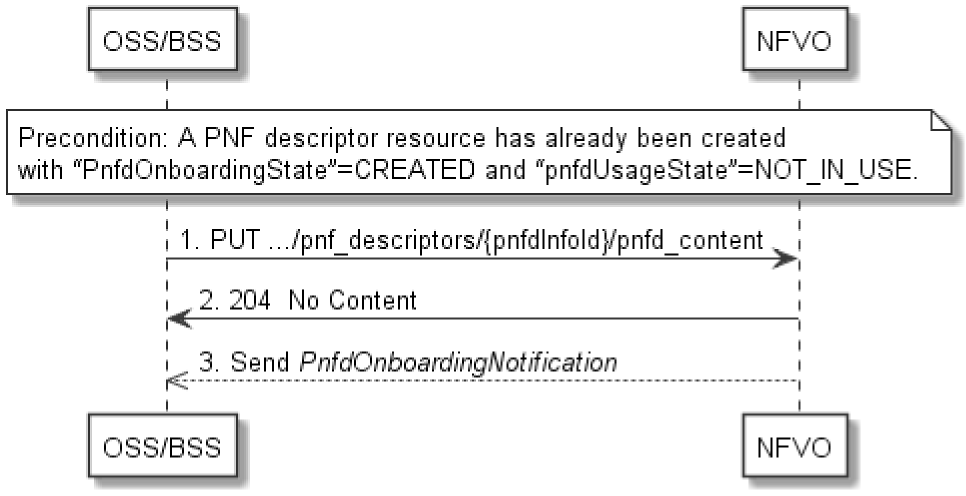
OSS/BSS
NFVO
Precondition: A PNF descriptor resource has already been created with "PnfdOnboardingState"=CREATED and "pnfdUsageState"=NOT_IN_USE.
1. PUT .../pnf_descriptors/{pnfdInfoId}/pnfd_content
2. 204 No Content
3. Send PnfdOnboardingNotification
OSS/BSS
NFVO

Figure 5.3.8-1: Flow of the uploading of PNFD content

Precondition: A PNF descriptor resource has already been created (i.e. "PnfdOnboardingState"=CREATED and "pnfdUsageState"=NOT_IN_USE).

The procedure consists of the following steps as illustrated in Figure 5.3.8-1:

1) The OSS/BSS sends a PUT request to a "PNFD Content" resource using a "Content-Type" HTTP header as defined in clause 5.4.4.3.3 of the present document.

2) The NFVO returns a "204 No Content" response to the OSS/BSS with an empty payload body for successful uploading of the PNFD content.

3) The NFVO sends a PnfdOnboardingNotification to the OSS/BSS.

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

5.3.9 Flow of the fetching of PNFD content

This clause describes the procedure for fetching the content of an onboarded PNFD.

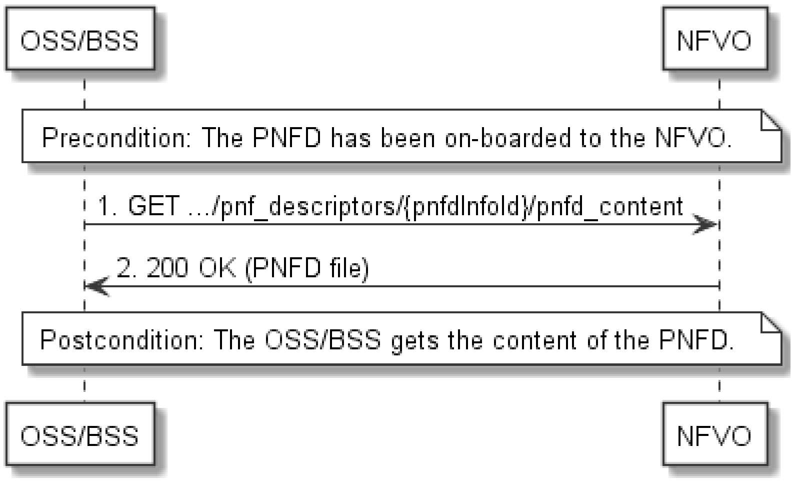
OSS/BSS
NFVO
Precondition: The PNFD has been on-boarded to the NFVO.
1. GET .../pnf_descriptors/{pnfdInfoId}/pnfd_content
2. 200 OK (PNFD file)
Postcondition: The OSS/BSS gets the content of the PNFD.
OSS/BSS
NFVO

Figure 5.3.9-1: Flow of the fetching of PNFD content

Precondition: The PNFD has been on-boarded to the NFVO.

Fetching an on-boarded PNFD, as illustrated in Figure 5.3.9-1, consists of the following steps.

1) The OSS/BSS sends a GET request to the "PNFD content" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the PNFD file in the payload body.

Postcondition: Upon successful completion, the OSS/BSS gets the content of the PNFD.

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

5.3.10 Flow of the deletion of an individual PNF descriptor resource

This clause describes the procedure for the deletion of an individual PNF descriptor resource.

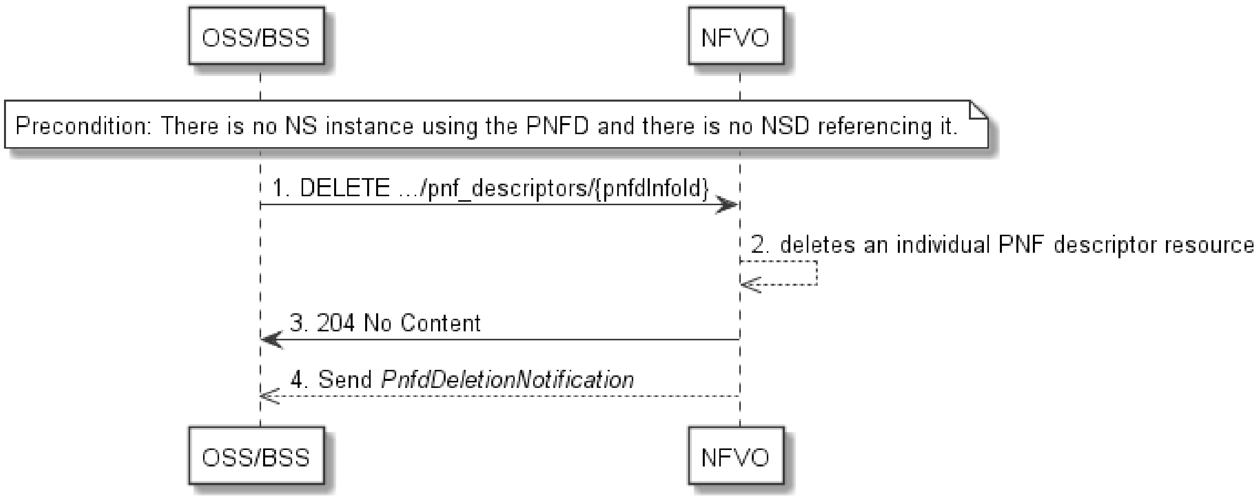
OSS/BSS
NFVO
Precondition: There is no NS instance using the PNFD and there is no NSD referencing it.
1. DELETE .../pnf_descriptors/{pnfdInfoId}
2. deletes an individual PNF descriptor resource
3. 204 No Content
4. Send PnfdDeletionNotification
OSS/BSS
NFVO

Figure 5.3.10-1: Flow of the deletion of an individual PNF descriptor resource

Precondition: NSD has been on boarded to the NFVO and in the Disabled state.

The procedure consists of the following steps as illustrated in Figure 5.3.10-1:

1) The OSS/BSS sends a DELETE request to an "Individual PNF descriptor" resource.

2) The NFVO deletes the individual PNF descriptor resource.

3) The NFVO returns a "204 No Content" response to the OSS/BSS with an empty payload body.

4) The NFVO sends to the OSS/BSS a PnfdDeletionNotification to indicate the deletion of the individual PNF descriptor resource.

5.3.11 Flow of the querying/reading of PNF descriptor resources

This clause describes the procedure for querying information about multiple PNF descriptor resources and reading information about an individual PNF descriptor resource.

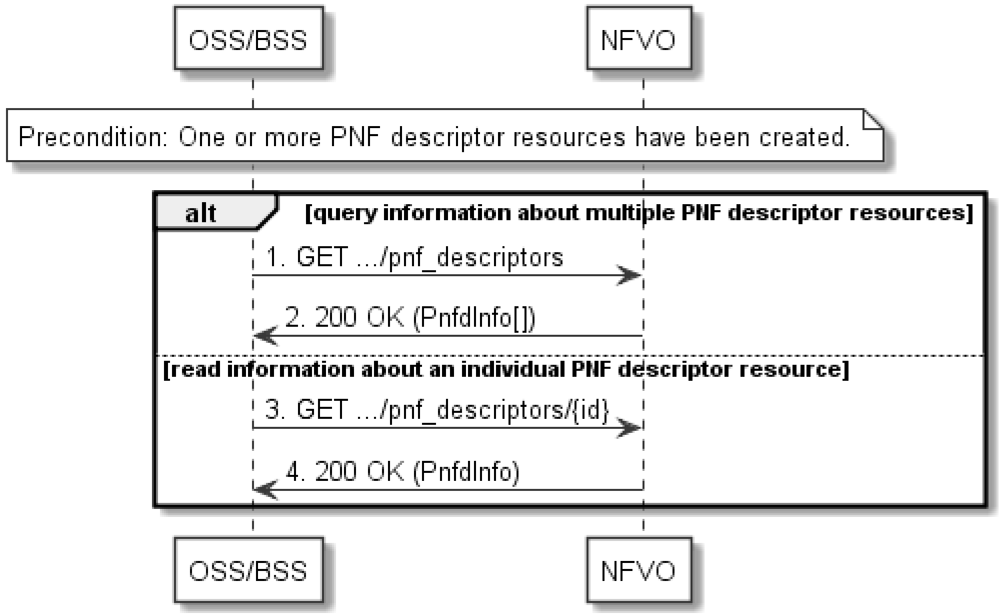
OSS/BSS
NFVO
Precondition: One or more PNF descriptor resources have been created.
alt
[query information about multiple PNF descriptor resources]
1. GET .../pnf_descriptors
2. 200 OK (PnfdInfo[])
[read information about an individual PNF descriptor resource]
3. GET .../pnf_descriptors/{id}
4. 200 OK (PnfdInfo)
OSS/BSS
NFVO

Figure 5.3.11-1: Flow of the querying/reading of PNF descriptor resources

The procedure consists of the following steps as illustrated in Figure 5.3.11-1:

Precondition: One or more PNF descriptor resources have been created:

1) If the OSS/BSS intends to query information about multiple PNF descriptor resources, it sends a GET request to the pnf_descriptors resource.

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

3) If the OSS/BSS intends to read information about an individual PNF descriptor resource, the OSS/BSS sends a GET request to the "Individual PNF descriptor" resource, addressed by the appropriate PnfdInfo identifier in its resource URI.

4) The NFVO returns a "200 OK" response, and includes in the payload body a data structure of type "PnfdInfo".

Postcondition: Upon successful completion, the OSS/BSS gets the information of multiple (i.e. zero or more) PNF descriptor resources or an individual PNF descriptor resource.

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

5.3.12 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to NSD management.

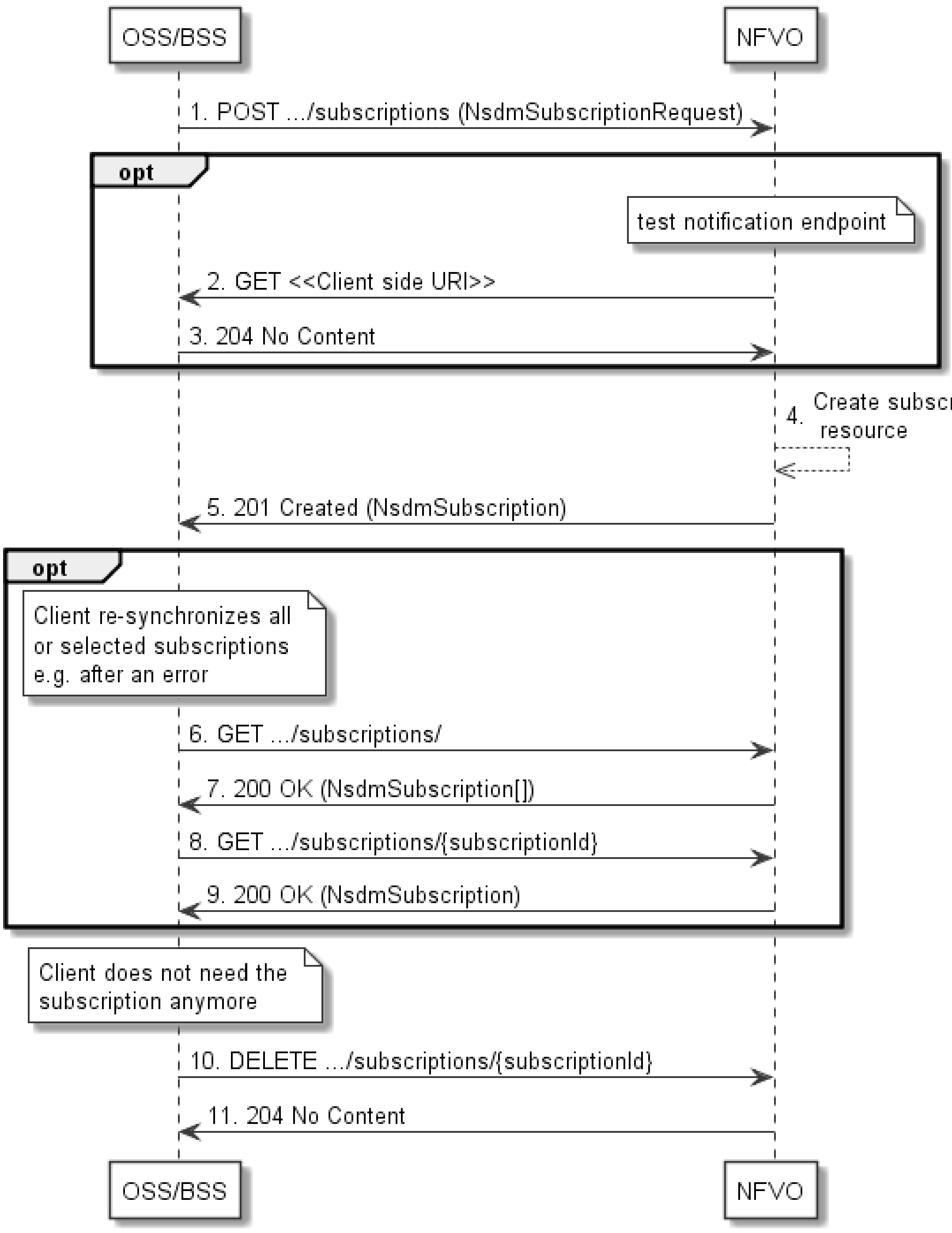
OSS/BSS
NFVO
1. POST .../subscriptions (NsdmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (NsdmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (NsdmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (NsdmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
OSS/BSS
NFVO

Figure 5.3.12-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in Figure 5.3.12-1:

1) The OSS/BSS sends a POST request to the "Subscriptions" resource including in the payload body a data structure of type "NsdmSubscriptionRequest". This data structure contains filtering criteria and a client side URI to which the NFVO will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the OSS/BSS as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

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

4) The NFVO creates a new subscription for notifications related to NS fault management, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "NsdmSubscription", representing the subscription resource just created by the NFVO, 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 OSS/BSS may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the NFVO.

8) Optionally, for example when trying to recover from an error situation, the OSS/BSS may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

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

10) When the OSS/BSS 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 NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

5.3.13 Flow of sending notifications

This clause describes the procedure for sending notifications related to NSD management.

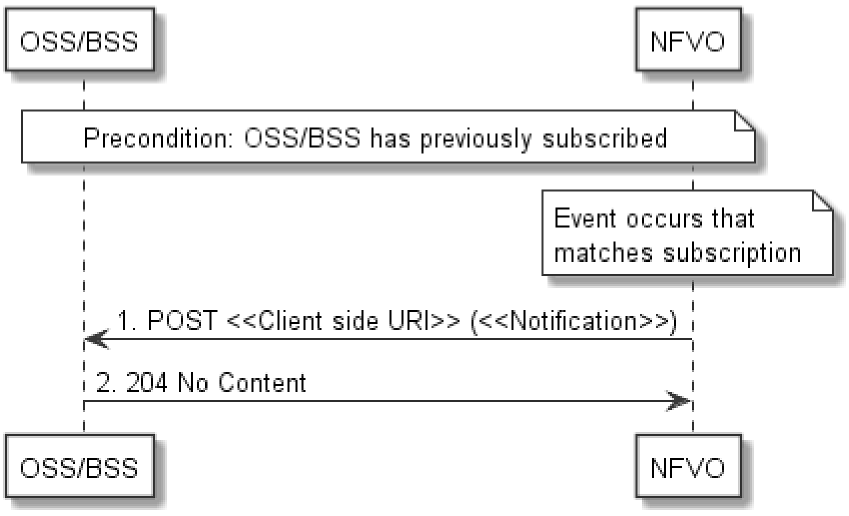
OSS/BSS
NFVO
Precondition: OSS/BSS has previously subscribed
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
OSS/BSS
NFVO

Figure 5.3.13-1: Flow of sending notifications

Precondition: The OSS/BSS has subscribed previously for notifications related to NSD management.

The procedure consists of the following steps as illustrated in Figure 5.3.13-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the NFVO 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 5.5.2.9, 5.5.2.10, 5.5.2.11 and 5.5.2.12).

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

Error handling: If the NFVO does not receive the "204 No Content" response from the OSS/BSS, it can retry sending the notification.

5.4 Resources

5.4.1 Introduction

This clause defines all the resource and methods provided by the NSD management interface.

The on-boarding of a NSD includes:

1) Creation of an individual NS descriptor resource
2) Uploading the NSD content
3) Validation of the NSD inside the NFVO

In the present document, the NSD is referred to as "on-boarded" only after these three procedures are successfully accomplished.

NOTE: Annex B describes the state model of NSD in the NFVO. It includes the state models for two phases, i.e. onboarding phase of NSD and operational phase of NSD.

Further, the on-boarding of a PNFD includes:

1) Creation of an individual PNF descriptor resource
2) Uploading the PNFD
3) Processing the PNFD, including validation, inside the NFVO

A PNFD is referred as "on-boarded" only after these three procedures are successfully accomplished.

5.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the NSD management interface.

5.4.2 Resource: NS Descriptors

5.4.2.1 Description

This resource represents NS descriptors. It can be used to create an individual NS descriptor resource, and to query multiple NS descriptor resources.

5.4.2.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/ns_descriptors

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 is used to create a new NS descriptor 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

	Data type	Cardinality	Description	
Request body	CreateNsdInfoRequest	1	Parameters of creating an NS descriptor resource, as defined in clause 5.5.2.4.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	NsdInfo	1	201 Created	An NS descriptor resource was created successfully, as a new NS descriptor resource. The response body shall contain a representation of the new NS descriptor resource, as defined in clause 5.5.2.2. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the new NS descriptor 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.

5.4.2.3.2 GET

The GET method queries information about multiple NS descriptor resources.

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 expression according to clause 4.3.2. The NFVO shall support receiving this filtering parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the NsdInfo and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO should support this parameter.

Name	Cardinality	Description
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 NsdInfo structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: userDefinedData.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

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**
	NsdInfo	0..N	200 OK	Information about zero or more NS descriptors. The response body shall contain in an array the representations of zero or more NS descriptors, as defined in clause 5.5.2.2. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

5.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3 Resource: Individual NS Descriptor

5.4.3.1 Description

This task resource represents an individual NS descriptor. The client can use this resource to modify, delete and read the information of the individual NS descriptor.

5.4.3.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/ns_descriptors/{nsdInfoId}

This resource shall support the resource URI variables 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.
nsdInfoId	Identifier of the individual NS descriptor resource. 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 NS descriptor 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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3.3.2 GET

The GET method reads information about an individual NS descriptor.

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			
Response body	Data type	Cardinality	Response Codes	Description
	NsdInfo	1	200 OK	Information about the individual NS descriptor. The response body shall contain a representation of the individual NS descriptor, 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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.3.3.4 PATCH

The PATCH method modifies the operational state and/or user defined data of an individual NS descriptor resource.

This method can be used to:

1) Enable a previously disabled individual NS descriptor resource, allowing again its use for instantiation of new network service with this descriptor. The usage state (i.e. "IN_USE/NOT_IN_USE") shall not change as a result.

2) Disable a previously enabled individual NS descriptor resource, preventing any further use for instantiation of new network service(s) with this descriptor. The usage state (i.e. "IN_USE/NOT_IN_USE") shall not change as a result.

3) Modify the user defined data of an individual NS descriptor 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	Response Codes	Description
Request body	NsdInfoModifications	1		Parameters for the modification of an individual NS descriptor resource, as defined in clause 5.5.2.1.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	NsdInfoModifications	1	200 OK	The operation was completed successfully. The response body shall contain attribute modifications for an 'Individual NS Descriptor' resource (see clause 5.5.2.6).
	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 an operational state mismatch, i.e. enable an already enabled or disable an already disabled individual NS descriptor resource, or the "nsdOnboardingState" is not ONBOARDED. 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.

5.4.3.3.5 DELETE

The DELETE method deletes an individual NS descriptor resource.

An individual NS descriptor resource can only be deleted when there is no NS instance using it (i.e. usageState = NOT_IN_USE) and has been disabled already (i.e. operationalState = DISABLED). Otherwise, the DELETE method shall fail.

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 operation has completed 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 the NS descriptor resource is in the enabled operational state (i.e. operationalState = ENABLED) or there are running NS instances using the concerned individual NS descriptor resource (i.e. usageState = IN_USE). 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: NSD Content

5.4.4.1 Description

This resource represents the content of the individual NS descriptor, i.e. NSD content. The client can use this resource to upload and download the content of the NSD.

5.4.4.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content

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.
nsdInfoId	Identifier of the individual NS descriptor. 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 NS descriptor 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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.4.3.2 GET

The GET method fetches the content of the NSD.

The NSD can be implemented as a single file or as a collection of multiple files. If the NSD is implemented in the form of multiple files, a ZIP file embedding these files shall be returned. If the NSD is implemented as a single file, either that file or a ZIP file embedding that file shall be returned.

The selection of the format is controlled by the "Accept" HTTP header passed in the GET request:

- If the "Accept" header contains only "text/plain" and the NSD is implemented as a single file, the file shall be returned; otherwise, an error message shall be returned.
- If the "Accept" header contains only "application/zip", the single file or the multiple files that make up the NSD shall be returned embedded in a ZIP file.
- If the "Accept" header contains both "text/plain" and "application/zip", it is up to the NFVO to choose the format to return for a single-file NSD; for a multi-file NSD, a ZIP file shall be returned.

NOTE: The structure of the NSD zip file is outside the scope of the present document.

This method shall follow the provisions specified in the Tables 5.4.4.3.2-1 and 5.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.4.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description
Request body	n/a		The request shall contain the appropriate entries in the "Accept" HTTP header as defined above. The request may contain a "Range" HTTP header to obtain single range of bytes from the NSD file. This can be used to continue an aborted transmission. If the NFVO does not support range requests, the NFVO shall ignore the 'Range" header, process the GET request, and return the whole NSD file with a 200 OK response (rather than returning a 4xx error status code).

	Data type	Cardinality	Response Codes	Description
Response body	n/a	1	200 OK	On success, the content of the NSD is returned. The payload body shall contain a copy of the file representing the NSD or a ZIP file that contains the file or multiple files representing the NSD, as specified above. The "Content-Type" HTTP header shall be set according to the format of the returned file, i.e. to "text/plain" for a YAML file or to "application/zip" for a ZIP file.
	n/a	1	206 Partial Content	On success, if the NFVO supports range requests, a single consecutive byte range from the content of the NSD file is returned. The response body shall contain the requested part of the NSD file. The "Content-Range" HTTP header shall be provided according to IETF RFC 7233 [23]. The "Content-Type" HTTP header shall be set as defined above for the "200 OK" response.
	ProblemDetails	0..1	406 Not AccepTable	If the "Accept" header does not contain at least one name of a content type for which the NFVO can provide a representation of the NSD, the NFVO shall respond with this response code. The "ProblemDetails" structure may be included with the "detail" attribute providing 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 "nsdOnboardingState" has a value different from ONBOARDED. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	416 Range Not Satisfiable	The byte range passed in the "Range" header did not match any available byte range in the NSD file (e.g. "access after end of file"). The response body may contain a ProblemDetails structure.
	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.

5.4.4.3.3 PUT

The PUT method is used to upload the content of a NSD.

The NSD to be uploaded can be implemented as a single file or as a collection of multiple files, as defined in clause 5.4.4.3.2. If the NSD is implemented in the form of multiple files, a ZIP file embedding these files shall be uploaded. If the NSD is implemented as a single file, either that file or a ZIP file embedding that file shall be uploaded.

The "Content-Type" HTTP header in the PUT request shall be set accordingly based on the format selection of the NSD:

- If the NSD to be uploaded is a text file, the "Content-Type" header is set to "text/plain".
- If the NSD to be uploaded is a zip file, the "Content-Type" header is set to "application/zip".

This method shall follow the provisions specified in the Tables 5.4.4.3.3-1 and 5.4.4.3.3-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.4.3.3-1: URI query parameters supported by the PUT method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.4.3.3-2: Details of the PUT request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a	1	The payload body contains a copy of the file representing the NSD or a ZIP file that contains the file or multiple files representing the NSD, as specified above. The request shall set the "Content-Type" HTTP header as defined above.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	202 Accepted	The NSD content was accepted for uploading, but the processing has not been completed. It is expected to take some time for processing (asynchronous mode). The response body shall be empty. See note.
	n/a	1	204 No Content	The NSD content was successfully uploaded and validated (synchronous mode). 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 NsdOnboardingState has a value other than CREATED. 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 re 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.
NOTE: The client can track the uploading progress by receiving the "NsdOnBoardingNotification" and "NsdOnBoardingFailureNotification" from the NFVO or by reading the status of the individual NS descriptor resource using the GET method.				

5.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5 Resource: PNF Descriptors

5.4.5.1 Description

This resource represents PNF descriptors and it can be used to create an individual PNF descriptor resource, and to query PNF descriptor resources.

5.4.5.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/pnf_descriptors

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

5.4.5.3 Resource methods

5.4.5.3.1 POST

The POST method is used to create a new PNF descriptor resource.

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

	Data type	Cardinality	Description	
Request body	CreatePnfdInfoRequest	1	Parameters of creating a PNF descriptor resource, as defined in clause 5.5.2.6.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PnfdInfo	1	201 Created	A PNF descriptor resource was created successfully, as a new PNF descriptor resource. The response body shall contain a representation of the new PNF descriptor resource, as defined in clause 5.5.2.5. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the new PNF descriptor 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.

5.4.5.3.2 GET

The GET method queries information about multiple PNF descriptor resources.

This method shall follow the provisions specified in the Tables 5.4.5.3.2-1 and 5.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this filtering parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the PnfdInfo and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO 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 NFVO shall support this parameter. The following attributes shall be excluded from the PnfdInfo structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: userDefinedData.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 5.4.5.3.2-2: 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	PnfdInfo	0..N	200 OK	Information about zero or more PNF descriptors. The response body shall contain in an array the representations of zero or more PNF descriptors, as defined in clause 5.5.2.2. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

5.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6 Resource: Individual PNF Descriptor

5.4.6.1 Description

This resource represents an individual PNF descriptor. The client can use this resource to modify, delete and read the information of the individual PNF descriptor resource.

5.4.6.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/pnf_descriptors/{pnfdInfoId}

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.
pnfdInfoId	Identifier of the individual PNF descriptor resource. 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 PNF descriptor 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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6.3.2 GET

The GET method reads information about an individual PNF descriptor.

This method shall follow the provisions specified in the Tables 5.4.6.3.2-1 and 5.4.6.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.6.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
None supported		

Table 5.4.6.3.2-2: 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
	PnfdInfo	1	200 OK	Information about the individual PNFD descriptor. The response body shall contain a representation of the individual PNF descriptor, as defined in clause 5.5.2.5.
	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.

5.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.6.3.4 PATCH

The PATCH method modifies the user defined data of an individual PNF descriptor resource.

This method shall follow the provisions specified in the Tables 5.4.6.3.4-1 and 5.4.6.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.6.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
None supported		

Table 5.4.6.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality		Description
Request body	PnfdInfoModifications	1	Parameters for the modification of an individual PNF descriptor resource, as defined in clause 5.5.2.4.	
Response body	Data type	Cardinality	Response Codes	Description
	PnfdInfoModifications	1	200 OK	The operation was completed successfully. The response body shall contain attribute modifications for an 'Individual PNF Descriptor' resource (see clause 5.5.2.4).
	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.

5.4.6.3.5 DELETE

The DELETE method deletes an individual PNF descriptor resource.

An individual PNF descriptor resource can only be deleted when there is no NS instance using it or there is NSD referencing it.

To delete all PNFD versions identified by a particular value of the "pnfdInvariantId" attribute, the procedure is to first use the GET method with filter "pnfdInvariantId" towards the PNF descriptors resource to find all versions of the PNFD. Then, the client uses the DELETE method described in this clause to delete each PNFD version individually.

This method shall follow the provisions specified in the Tables 5.4.6.3.5-1 and 5.4.6.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.6.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.6.3.5-2: 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 operation has completed 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.

5.4.7 Resource: PNFD Content

5.4.7.1 Description

This resource represents the content of the individual PNF descriptor, i.e. PNFD content. The client can use this resource to upload and download the content of the PNFD.

5.4.7.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/pnf_descriptors/{pnfdInfoId}/pnfd_content

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.
pnfdInfoId	Identifier of the individual PNF descriptor. 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 PNF descriptor 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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7.3.2 GET

The GET method fetches the content of the PNFD.

This method shall follow the provisions specified in the Tables 5.4.7.3.2-1 and 5.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	**Cardinality**	**Description**
none supported		

Table 5.4.7.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**		**Description**
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, the content of the PNFD is returned. The payload body shall contain a copy of the file representing the PNFD. The "Content-Type" HTTP header shall be set to "text/plain".
	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 pnfdOnboardingState has a value different from ONBOARDED. 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.

5.4.7.3.3 PUT

The PUT method is used to upload the content of a PNFD.

This method shall follow the provisions specified in the Tables 5.4.7.3.3-1 and 5.4.7.3.3-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.7.3.3-1: URI query parameters supported by the PUT method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.7.3.3-2: Details of the PUT request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	n/a	1		The payload body contains a copy of the file representing the PNFD. The request shall set the "Content-Type" HTTP header to "text/plain".
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	204 No Content	The PNFD content was successfully uploaded and validated. 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 PnfdOnboardingState has a value other than CREATED. 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.

5.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8 Resource: Subscriptions

5.4.8.1 Description

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

5.4.8.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/subscriptions

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.

5.4.8.3 Resource methods

5.4.8.3.1 POST

The POST method creates a new subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 5.4.8.3.1-1 and 5.4.8.3.1-2.

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 OSS, and might make sense only in very rare use cases. Consequently, the NFVO 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 5.4.8.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
n/a		

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

	Data type	Cardinality		Remarks
Request body	NsdmSubscriptionRequest	1	Details of the subscription to be created, as defined in clause 5.5.2.7.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	NsdmSubscription	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 exits and the policy of the NFVO 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.

5.4.8.3.2 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 support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 5.4.8.3.2-1 and 5.4.8.3.2-2.

Table 5.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
filter	0..1	Attribute filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the NsdmSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 5.4.8.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Remarks
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	NsdmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e., zero or more representations of NSD management subscriptions as defined in clause 5.5.2.8. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails		400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

5.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.8.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9 Resource: Individual subscription

5.4.9.1 Description

This resource represents an individual subscription. It can be used by the client to read and to terminate a subscription to notifications related to NSD management.

5.4.9.2 Resource definition

The resource URI is:

{apiRoot}/nsd/v1/subscriptions/{subscriptionId}

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.
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.	

5.4.9.3 Resource methods

5.4.9.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.2 GET

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

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 5.4.9.3.2-1 and 5.4.9.3.2-2.

Table 5.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
n/a		

Table 5.4.9.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	NsdmSubscription	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.

5.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.9.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 5.4.9.3.5-1 and 5.4.9.3.5-2.

Table 5.4.9.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Remarks
n/a		

Table 5.4.9.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	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.

5.4.10 Resource: Notification endpoint

5.4.10.1 Description

This resource represents a notification endpoint. The server can use this resource to send notifications to a subscribed client, which has provided the URI of this resource during the subscription process.

5.4.10.2 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 5.4.10.2-1.

Table 5.4.10.2-1: Resource URI variables for this resource

Name	Definition
n/a	

5.4.10.3 Resource methods

5.4.10.3.1 POST

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

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 5.4.10.3.1-1 and 5.4.10.3.1-2.

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

Name	Cardinality	Remarks
n/a		

Each notification request body shall include exactly one of the alternatives defined in Table 5.4.10.3.1-2.

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

	Data type	Cardinality	Remarks	
Request body	NsdOnBoardingNotification	1	A notification about the successful on-boarding of an NSD.	
	NsdOnBoardingFailureNotification	1	A notification about the failure of on-boarding an NSD.	
	NsdChangeNotification	1	A notification about the state change of an on-boarded NSD.	
	NsdDeletionNotification	1	A notification about the deletion of an on-boarded NSD.	
	PnfdOnBoardingNotification	1	A notification about the successful on-boarding of a PNFD.	
	PnfdOnBoardingFailure Notification	1	A notification about the failure of on-boarding a PNFD.	
	PnfdDeletionNotification	1	A notification about the deletion of an on-boarded PNFD.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	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.

5.4.10.3.2 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 5.4.10.3.2-1 and 5.4.10.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 5.4.10.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 5.4.10.3.2-2: 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.

5.4.10.3.3 PUT

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10.3.4 PATCH

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.4.10.3.5 DELETE

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

5.5 Data model

5.5.1 Introduction

This clause defines the request and response data structures of the NSD Lifecycle management interface.

5.5.2 Resource and notification data types

5.5.2.1 Type: NsdInfoModifications

This type represents attribute modifications for an individual NS descriptor resource based on the "NsdInfo" data type. The attributes of "NsdInfo" that can be modified are included in the "NsdInfoModifications" data type.

The "NsdInfoModifications" data type shall comply with the provisions defined in Table 5.5.2.1-1.

Table 5.5.2.1-1: Definition of the NsdInfoModifications data type

Attribute name	Data type	Cardinality	Description
sdOperationalState	NsdOperationalStateType	0..1	New value of the "nsdOperationalState" attribute in "NsdInfo" data type. See note. Permitted values: ENABLED DISABLED
userDefinedData	KeyValuePairs	0..1	Modifications of the "userDefinedData" attribute in "NsdInfo" data type. See note. If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [25].
NOTE: At least one of the attributes - nsdOperationalState and userDefinedData - shall be present.			

5.5.2.2 Type: NsdInfo

This type represents a response for the query NSD operation. It shall comply with the provisions defined in Table 5.5.2.2-1.

Table 5.5.2.2-1: Definition of the NsdInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the onboarded individual NS descriptor resource. This identifier is allocated by the NFVO.
nsdId	Identifier	0..1	This identifier, which is allocated by the NSD designer, identifies the NSD in a globally unique way. It is copied from the NSD content and shall be present after the NSD content is on-boarded.
nsdName	String	0..1	Name of the onboarded NSD. This information is copied from the NSD content and shall be present after the NSD content is on-boarded.
nsdVersion	Version	0..1	Version of the on-boarded NSD. This information is copied from the NSD content and shall be present after the NSD content is on-boarded.
nsdDesigner	String	0..1	Designer of the on-boarded NSD. This information is copied from the NSD content and shall be present after the NSD content is on-boarded.
nsdInvariantId	Identifier	0..1	This identifier, which is allocated by the NSD designer, identifies an NSD in a version independent manner. This information is copied from the NSD content and shall be present after the NSD content is on-boarded.
vnfPkgIds	Identifier	0..N	Identifies the VNF package for the VNFD referenced by the on-boarded NS descriptor resource. See note 1.
pnfdInfoIds	Identifier	0..N	Identifies the PnfdInfo element for the PNFD referenced by the on-boarded NS descriptor resource.
nestedNsdInfoIds	Identifier	0..N	Identifies the NsdInfo element for the nested NSD referenced by the on-boarded NS descriptor resource. See note 1.
nsdOnboardingState	NsdOnboardingState Type	1	Onboarding state of the individual NS descriptor resource. See note 4.
onboardingFailureDetails	ProblemDetails	0..1	Failure details of current onboarding procedure. See clause 4.3.5.3 for the details of "ProblemDetails" structure. It shall be present when the "nsdOnboardingState" attribute is CREATED and the uploading or processing fails in NFVO.
nsdOperationalState	NsdOperationalState Type	1	Operational state of the individual NS descriptor resource. This attribute can be modified with the PATCH method. See notes 2 and 4.
nsdUsageState	NsdUsageStateType	1	Usage state of the individual NS descriptor resource. See notes 3 and 4.
userDefinedData	KeyValuePairs	0..1	User defined data for the individual NS descriptor resource. This attribute can be modified with the PATCH method.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>nsd_content	Link	1	Link to the NSD content resource.
NOTE 1: At least one of the attributes - vnfPkgId and nestedNsdInfoId shall be present, after the NSD is on-boarded. NOTE 2: If the value of the nsdOnboardingState attribute is not equal to "ONBOARDED", the value of the nsdOperationalState attribute shall be equal to "DISABLED". NOTE 3: If the value of the nsdOnboardingState attribute is not equal to "ONBOARDED", the value of the nsdUsageState attribute shall be equal to "NOT_IN_USE". NOTE 4: State changes of a NSD are illustrated in clause B.2.			

5.5.2.3 Type: CreateNsdInfoRequest

This type creates a completely new NS descriptor resource. It shall comply with the provisions defined in Table 5.5.2.3-1.

Table 5.5.2.3-1: Definition of the CreateNsdInfoRequest data type

Attribute name	Data type	Cardinality	Description
userDefinedData	KeyValuePairs	0..1	User-defined data for the NS descriptor resource to be created. It shall be present when the user defined data is set for the individual NS descriptor resource to be created.

5.5.2.4 Type: PnfdInfoModifications

This type represents attribute modifications for an individual PNF descriptor resource based on the "PnfdInfo" data type. The attributes of "PnfdInfo" that can be modified are included in the "PnfdInfoModifications" data type.

The "PnfdInfoModifications" data type shall comply with the provisions defined in Table 5.5.2.4-1.

Table 5.5.2.4-1: Definition of the PnfdInfoModifications data type

Attribute name	Data type	Cardinality	Description
userDefinedData	KeyValuePairs	1	Modifications of the "userDefinedData" attribute in "PnfdInfo" data type. If present, these modifications shall be applied according to the rules of JSON Merge PATCH (see IETF RFC 7396 [25]).

5.5.2.5 Type: PnfdInfo

This type represents a response for the query PNFD operation. It shall comply with the provisions defined in Table 5.5.2.5-1.

Table 5.5.2.5-1: Definition of the PnfdInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the onboarded individual PNF descriptor resource. This identifier is allocated by the NFVO.
pnfdId	Identifier	0..1	This identifier, which is managed by the PNFD designer, identifies the PNFD in a globally unique way. It is copied from the PNFD content and shall be present after the PNFD content is on-boarded.
pnfdName	String	0..1	Name of the onboarded PNFD. This information is copied from the PNFD content and shall be present after the PNFD content is on-boarded.
pnfdVersion	Version	0..1	Version of the onboarded PNFD. This information is copied from the PNFD content and shall be present after the PNFD content is on-boarded.
pnfdProvider	String	0..1	Provider of the onboarded PNFD. This information is copied from the PNFD content and shall be present after the PNFD content is on-boarded.
pnfdInvariantId	Identifier	0..1	Identifies a PNFD in a version independent manner. This attribute is invariant across versions of PNFD.
pnfdOnboardingState	PnfdOnboardingStateType	1	Onboarding state of the individual PNF descriptor resource.
onboardingFailureDetails	ProblemDetails	0..1	Failure details of current onboarding procedure. See clause 4.3.5.3 for the details of "ProblemDetails" structure. It shall be present when the "pnfdOnboardingState" attribute is CREATED and the uploading or processing fails in the NFVO.
pnfdUsageState	PnfdUsageStateType	1	Usage state of the individual PNF descriptor resource.
userDefinedData	KeyValuePairs	0..1	User defined data for the individual PNF descriptor resource. This attribute can be modified with the PATCH method.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>pnfd_content	Link	1	Link to the PNFD Content resource.

5.5.2.6 Type: CreatePnfdInfoRequest

This type creates a new PNF descriptor resource. It shall comply with the provisions defined in Table 5.5.2.6-1.

Table 5.5.2.6-1: Definition of the CreatePnfdInfoRequest data type

Attribute name	Data type	Cardinality	Description
userDefinedData	KeyValuePairs	0..1	User-defined data for the PNF descriptor resource to be created. It shall be present when the user defined data is set for the individual PNF descriptor resource to be created.

5.5.2.7 Type: NsdmSubscriptionRequest

This type represents a subscription request related to notifications about NSD management. It shall comply with the provisions defined in Table 5.5.2.7-1.

Table 5.5.2.7-1: Definition of the NsdmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	NsdmNotificationsFilter	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.

5.5.2.8 Type: NsdmSubscription

This type represents a subscription related to notifications about NSD management. It shall comply with the provisions defined in Table 5.5.2.8-1.

Table 5.5.2.8-1: Definition of the NsdmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	NsdmNotificationsFilter	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.

5.5.2.9 Type: NsdOnboardingNotification

This type represents an NSD management notification, which informs the receiver of the successful on-boarding of an NSD. It shall comply with the provisions defined in Table 5.5.2.9-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when the "nsdOnboardingState" attribute of a new NSD has changed to "ONBOARDED".

Table 5.5.2.9-1: Definition of the NsdOnboardingNotification 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 "NsdOnboardingNotification" 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.
nsdInfoId	Identifier	1	Identifier of the NSD information object. This identifier is allocated by the NFVO.
nsdId	Identifier	1	This identifier, which is managed by the service provider, identifies the NSD in a globally unique way. It is copied from the on-boarded NSD.
_links	NsdmLinks	1	Links to resources related to this notification.

5.5.2.10 Type: NsdOnboardingFailureNotification

This type represents an NSD management notification, which informs the receiver of the failure of on-boarding an NSD. It shall comply with the provisions defined in Table 5.5.2.10-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when the on-boarding of an NSD has failed.

Table 5.5.2.10-1: Definition of the NsdOnboardingFailureNotification 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 "NsdOnboardingFailureNotification" 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.
nsdInfoId	Identifier	1	Identifier of the NSD information object. This identifier is allocated by the NFVO.
nsdId	Identifier	0..1	This identifier, which is managed by the service provider, identifies the NSD in a globally unique way.
onboardingFailureDetails	ProblemDetails	1	Failure details of current onboarding procedure. See clause 4.3.5.3 for the details of "ProblemDetails" structure.
_links	NsdmLinks	1	Links to resources related to this notification.

5.5.2.11 Type: NsdChangeNotification

This type represents an NSD management notification, which informs the receiver of a change of the "nsdOperationalState" attribute of an on-boarded NSD. Changes in the value of the "nsdUsageState" and "nsdOnboardingState" attributes are not reported. The notification shall comply with the provisions defined in Table 5.5.2.11-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when the value of the "nsdOperationalState" attribute has changed, and the "nsdOperationalState" attribute has the value "ONBOARDED".

Table 5.5.2.11-1: Definition of the NsdChangeNotification 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 "NsdChangeNotification" 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.
nsdInfoId	Identifier	1	Identifier of the NSD information object. This identifier is allocated by the NFVO.
nsdId	Identifier	1	This identifier, which is managed by the service provider, identifies the NSD in a globally unique way. It is copied from the on-boarded NSD.
nsdOperationalState	NsdOperationalStateType	1	New operational state of the on-boarded NSD.
_links	NsdmLinks	1	Links to resources related to this notification.

5.5.2.12 Type: NsdDeletionNotification

This type represents an NSD management notification, which informs the receiver of the deletion of an on-boarded NSD. The notification shall comply with the provisions defined in Table 5.5.2.12-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when it has deleted an on-boarded NSD.

Table 5.5.2.12-1: Definition of the NsdDeletionNotification 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 "NsdDeletionNotification " 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.
nsdInfoId	Identifier	1	Identifier of the NSD information object. This identifier is allocated by the NFVO.
nsdId	Identifier	1	This identifier, which is managed by the service provider, identifies the NSD in a globally unique way. It is copied from the on-boarded NSD.
_links	NsdmLinks	1	Links to resources related to this notification.

5.5.2.13 Type: PnfdOnboardingNotification

This type represents a PNFD management notification, which informs the receiver of the successful on-boarding of a PNFD. It shall comply with the provisions defined in Table 5.5.2.13-1. The support of this notification is mandatory.

The notification is triggered when a new PNFD is on-boarded.

Table 5.5.2.13-1: Definition of the PnfdOnboardingNotification 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 "PnfdOnboardingNotification" 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.
pnfdInfoId	Identifier	1	Identifier of the PNFD information object. This identifier is allocated by the NFVO.
pnfdId	Identifier	1	This identifier, which is managed by the service provider, identifies the PNFD in a globally unique way. It is copied from the on-boarded PNFD.
_links	PnfdmLinks	1	Links to resources related to this notification.

5.5.2.14 Type: PnfdOnboardingFailureNotification

This type represents a PNFD management notification, which informs the receiver of the failure of on-boarding a PNFD. It shall comply with the provisions defined in Table 5.5.2.14-1. The support of this notification is mandatory.

The notification is triggered when the on-boarding of a PNFD fails.

Table 5.5.2.14-1: Definition of the PnfdOnboardingFailureNotification 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 "PnfdOnboardingFailureNotification" 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.
pnfdInfoId	Identifier	1	Identifier of the PNFD information object. This identifier is allocated by the NFVO.
pnfdId	Identifier	0..1	This identifier, which is managed by the service provider, identifies the PNFD in a globally unique way.
onboardingFailureDetails	ProblemDetails	1	Failure details of current onboarding procedure. See clause 4.3.5.3 for the details of "ProblemDetails" structure.
_links	PnfdmLinks	1	Links to resources related to this notification.

5.5.2.15 Type: PnfdDeletionNotification

This type represents a PNFD management notification, which informs the receiver of the deletion of an on-boarded PNFD. The notification shall comply with the provisions defined in Table 5.5.2.15-1. The support of this notification is mandatory.

The notification is triggered when an on-boarded PNFD is deleted.

Table 5.5.2.15-1: Definition of the PnfdDeletionNotification 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 "PnfdDeletionNotification " 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.
pnfdInfoId	Identifier	1	Identifier of the PNFD information object. This identifier is allocated by the NFVO.
pnfdId	Identifier	1	This identifier, which is managed by the service provider, identifies the PNFD in a globally unique way. It is copied from the on-boarded PNFD.
_links	PnfdmLinks	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: NsdmNotificationsFilter

This type represents a subscription filter related to notifications about NSD management. It shall comply with the provisions defined in Table 5.5.3.2-1.

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 5.5.3.2-1: Definition of the NsdmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: NsdOnBoardingNotification NsdOnboardingFailureNotification NsdChangeNotification NsdDeletionNotification PnfdOnBoardingNotification PnfdOnBoardingFailureNotification PnfdDeletionNotification See note 1.
nsdInfoId	Identifier	0..N	Match the NsdInfo identifier which is allocated by the NFVO. See note 2.
nsdId	Identifier	0..N	Match the NSD identifier, which is allocated by the NSD designer. See note 2.
nsdName	String	0..N	Match the name of the onboarded NSD.
nsdVersion	Version	0..N	Match the NSD version listed as part of this attribute.
nsdDesigner	String	0..N	Match the NSD designer of the on-boarded NSD.
nsdInvariantId	Identifier	0..N	Match the NSD invariant identifier which is allocated by the NSD designer and identifies an NSD in a version independent manner.
vnfPkgIds	Identifier	0..N	Match VNF packages with a package identifier listed in the attribute.
pnfdInfoIds	Identifier	0..N	Match the PnfdInfo identifier for the PNFD referenced by the on-boarded NSD. See note 3.
nestedNsdInfoIds	Identifier	0..N	Match the NsdInfo identifier for the nested NSD referenced by the on-boarded NSD.
nsdOnboardingState	NsdOnboardingStateType	0..N	Match particular on-boarding state of the NSD.
nsdOperationalState	NsdOperationalStateType	0..N	Match particular operational state of the on-boarded NSD.
nsdUsageState	NsdUsageStateType	0..N	Match particular usage state of the on-boarded NSD.
pnfdId	Identifier	0..N	Match the PNFD identifier which is copied from the PNFD content. See note 3.
pnfdName	String	0..N	Match the name of the onboarded PNFD.
pnfdVersion	Version	0..N	Match the PNFD designer of the on-boarded PNFD.
pnfdProvider	String	0..N	Match the provider of the on-boarded PNFD.
pnfdInvariantId	Identifier	0..N	Match the PNFD in a version independent manner.
pnfdOnboardingState	PnfdOnboardingStateType	0..N	Match particular onboarding state of the PNFD.
pnfdUsageState	PnfdUsageStateType	0..N	Match the usage state of the individual PNF descriptor resource.
NOTE 1: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems. NOTE 2: The attributes "nsdId" and "nsdInfoId" are alternatives to reference to a particular NSD in a filter. They should not be used both in the same filter instance, but one alternative should be chosen. NOTE 3: The attributes "pnfdId" and "pnfdInfoId" are alternatives to reference to a particular PNFD in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

5.5.3.3 Type: NsdmLinks

This type represents the links to resources that an NSD management notification can contain. It shall comply with the provisions defined in Table 5.5.3.3-1.

Table 5.5.3.3-1: Definition of the NsdmLinks data type

Attribute name	Data type	Cardinality	Description
nsdInfo	NotificationLink	1	Link to the resource representing the NSD to which the notified change applies, i.e. the individual NS descriptor resource that represents the NSD.
subscription	NotificationLink	1	Link to the related subscription.

5.5.3.4 Type: PnfdmLinks

This type represents the links to resources that a PNFD management notification can contain. It shall comply with the provisions defined in Table 5.5.3.4-1.

Table 5.5.3.4-1: Definition of the PnfdmLinks data type

Attribute name	Data type	Cardinality	Description
pnfdInfo	NotificationLink	1	Link to the resource representing the PNFD to which the notified change applies, i.e. the individual PNF descriptor resource that represents the PNFD.
subscription	NotificationLink	1	Link to the related subscription.

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 4.4.2.

5.5.4.3 Enumeration: NsdOperationalStateType

The enumeration NsdOperationalStateType shall comply with the provisions defined in Table 5.5.4.3-1. It indicates the operational state of the resource.

Table 5.5.4.3-1: Enumeration NsdOperationalStateType

Enumeration value	Description
ENABLED	The operational state of the resource is enabled.
DISABLED	The operational state of the resource is disabled.

5.5.4.4 Enumeration: NsdUsageStateType

The enumeration NsdUsageStateType shall comply with the provisions defined in Table 5.5.4.4-1. It indicates the usage state of the resource.

Table 5.5.4.4-1: Enumeration NsdUsageStateType

Enumeration value	Description
IN_USE	The resource is in use.
NOT_IN_USE	The resource is not-in-use.

5.5.4.5 Enumeration: NsdOnboardingStateType

The enumeration NsdOnboardingStateType shall comply with the provisions defined in Table 5.5.4.5-1. It indicates the onboarding state of the NSD.

Table 5.5.4.5-1: Enumeration NsdOnboardingStateType

Enumeration value	Description
CREATED	The NSD information object is created.
UPLOADING	The associated NSD content is being uploaded.
PROCESSING	The associated NSD content is being processed, e.g. validation.
ONBOARDED	The associated NSD content is on-boarded.

5.5.4.6 Enumeration: PnfdOnboardingStateType

The enumeration PnfdOnboardingStateType shall comply with the provisions defined in Table 5.5.4.6-1. It indicates the onboarding state of the individual PNF descriptor resource.

Table 5.5.4.6-1: Enumeration PnfdOnboardingStateType

Enumeration value	Description
CREATED	The PNF descriptor resource is created.
UPLOADING	The associated PNFD content is being uploaded.
PROCESSING	The associated PNFD content is being processed, e.g. validation.
ONBOARDED	The associated PNFD content is on-boarded.

5.5.4.7 Enumeration: PnfdUsageStateType

The enumeration PnfdUsageStateType shall comply with the provisions defined in Table 5.5.4.7-1. It indicates the usage state of the resource.

Table 5.5.4.7-1: Enumeration PnfdUsageStateType

Enumeration value	Description
IN_USE	The resource is in use.
NOT_IN_USE	The resource is not-in-use.

6 NS Lifecycle Management interface

6.1 Description

This interface allows the OSS/BSS to invoke NS lifecycle management operations of NS instances towards the NFVO, and to subscribe to notifications regarding NS lifecycle changes provided by the NFVO.

The operations provided through this interface are as follows:

- Create NS Identifier
- Instantiate NS
- Scale NS
- Update NS
- Query NS
- Terminate NS

- Delete NS Identifier
- Heal NS
- Get Operation Status
- Subscribe
- Query Subscription Information
- Notify
- Terminate Subscription

6.1a API version

For the NS lifecycle management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 1, and the PATCH version number shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.4.1 of the present document.

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 "nslcm" shall be used to represent {apiName}. 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 NS lifecycle management interface.

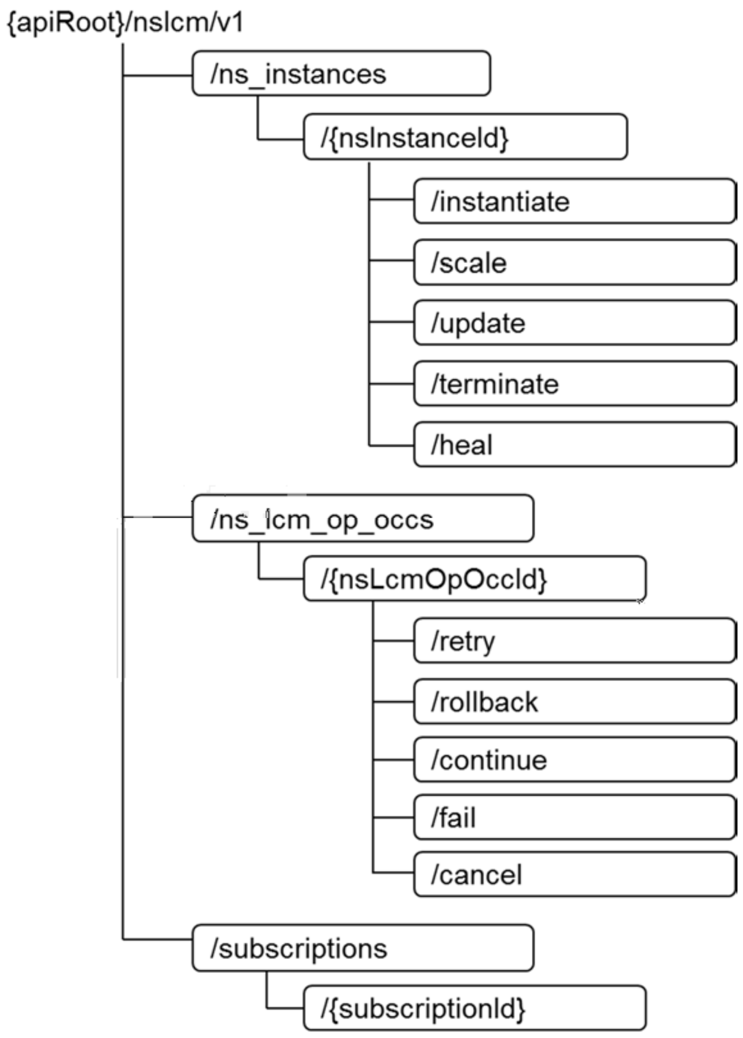

Figure 6.2-1: Resource URI structure of NS Lifecycle Management Interface

Table 6.2-1 lists the individual resources defined, and the applicable HTTP methods. The NFVO shall support responding to requests for all HTTP methods on the resources in Table 6.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 6.2-1: Resources and methods overview of the NS Lifecycle Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
NS instances	/ns_instances	GET	M	Query multiple NS instances
		POST	M	Create a NS instance resource
Individual NS instance	/ns_instances/{nsInstanceId}	GET	M	Read an individual NS instance resource
		DELETE	M	Delete NS instance resource
Instantiate NS task	/ns_instances/{nsInstanceId}/instantiate	POST	M	Instantiate a NS
Scale NS task	/ns_instances/{nsInstanceId}/scale	POST	M	Scale a NS instance
Update NS task	/ns_instances/{nsInstanceId}/update	POST	M	Updates a NS instance
Terminate NS task	/ns_instances/{nsInstanceId}/terminate	POST	M	Terminate a NS instance
Heal NS task	/ns_instances/{nsInstanceId}/heal	POST	M	Heal a NS instance
NS LCM operation occurrences	/ns_lcm_op_ops	GET	M	Query multiple NS LCM operation occurrences
Individual NS LCM operation occurrence	/ns_lcm_op_ops/{nsLcmOpOccId}	GET	M	Read an individual NS LCM operation occurrence resource
Retry operation task	/ns_lcm_op_occs/{nsLcmOpOccId}/retry	POST	M	Retry a NS lifecycle management operation occurrence
Rollback operation task	/ns_lcm_op_occs/{nsLcmOpOccId}/rollback	POST	M	Rollback a NS lifecycle management operation occurrence
Continue operation task	/ns_lcm_op_occs/{nsLcmOpOccId}/continue	POST	M	Continue a NS lifecycle management operation occurrence
Fail operation task	/ns_lcm_op_occs/{nsLcmOpOccId}/fail	POST	M	Mark a NS lifecycle management operation occurrence as failed
Cancel operation task	/ns_lcm_op_occs/{nsLcmOpOccId}/cancel	POST	M	Cancel a NS lifecycle management operation occurrence
Subscriptions	/subscriptions	POST	M	Subscribe to NS 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	See note	Notify about NS lifecycle change. See note
		GET	See note	Test the notification endpoint. See note
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the OSS/BSS. If the OSS/BSS supports invoking the POST method on the "Subscriptions" resource towards the NFVO, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

6.3 Sequence diagrams (informative)

6.3.1 Flow of the creation of a NS instance resource

This clause describes the procedure for the creation of a NS instance resource.

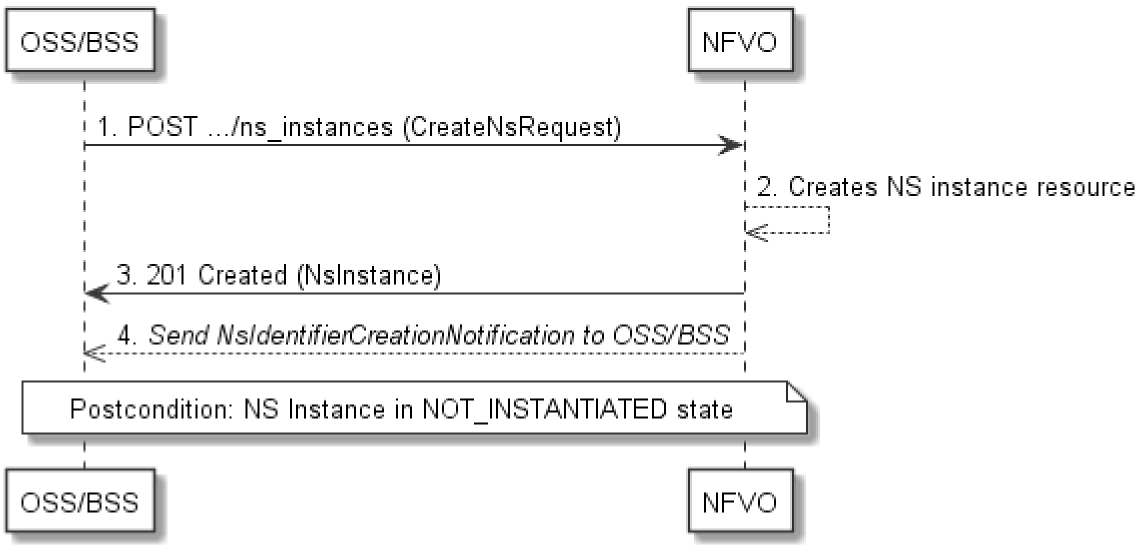
OSS/BSS
NFVO
1. POST .../ns_instances (CreateNsRequest)
2. Creates NS instance resource
3. 201 Created (NsInstance)
4. Send NsIdentifierCreationNotification to OSS/BSS
Postcondition: NS Instance in NOT_INSTANTIATED state
OSS/BSS
NFVO

Figure 6.3.1-1: Flow of the creation of a NS instance resource

NOTE: Due to possible race conditions, the 201 response and the NsIdentifierCreationNotification can arrive in any order at the OSS/BSS.

The procedure consists of the following steps as illustrated in Figure 6.3.1-1:

1) The OSS/BSS sends a POST request to the "NS Instances" resource including in the entity body a data structure of type "CreateNsRequest".

2) The NFVO creates a new NS instance resource in NOT_INSTANTIATED state, and the associated NS instance identifier.

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

4) The NFVO sends a NsIdentifierCreationNotification to the OSS/BSS to indicate the creation of the NS instance resource and the associated NS instance identifier.

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

6.3.2 Flow of the deletion of a NS instance resource

This clause describes the procedure for the deletion of a NS instance resource.

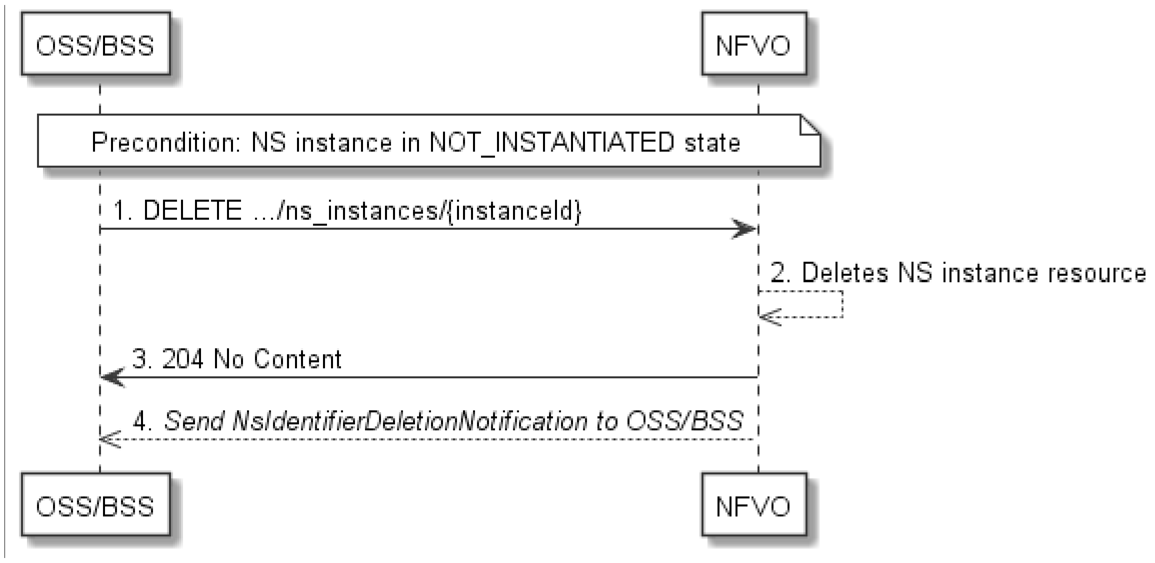
OSS/BSS
NFVO
Precondition: NS instance in NOT_INSTANTIATED state
1. DELETE .../ns_instances/{instanceId}
2. Deletes NS instance resource
3. 204 No Content
4. Send NsIdentifierDeletionNotification to OSS/BSS
OSS/BSS
NFVO

Figure 6.3.2-1: Flow of the deletion of a NS instance resource

NOTE: Due to possible race conditions, the 204 response and the NsIdentifierDeletionNotification can arrive in any order at the OSS/BSS.

Precondition: The resource representing the NS instance to be deleted is in NOT_INSTANTIATED state.

The procedure consists of the following steps as illustrated in Figure 6.3.2-1:

1) The OSS/BSS sends a DELETE request to the "Individual NS Instance" resource.

2) The NFVO deletes the NS instance resource and the associated NS instance identifier.

3) The NFVO returns a "204 No Content" response with an empty entity body.

4) The NFVO sends to the OSS/BSS a NsIdentifierDeletionNotification to indicate the deletion of the NS instance resource and the associated NS instance identifier.

Error Handling: If the NS instance is not in NOT_INSTANTIATED state, the NFVO rejects the deletion request.

6.3.3 Flow of NS lifecycle management operations triggered by task resources

This clause describes the general sequence for NS Lifecycle Management operations that operate on a NS 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 Table 6.3.3-1.

This flow is applicable to the following operations:

- Instantiate NS
- Scale NS
- Update NS
- Heal NS
- Terminate NS

Figure 6.3.3-1 illustrates the general lifecycle management flow. Placeholders in this flow allow for differentiating between the operations and are marked with double angular brackets "<<…>>".

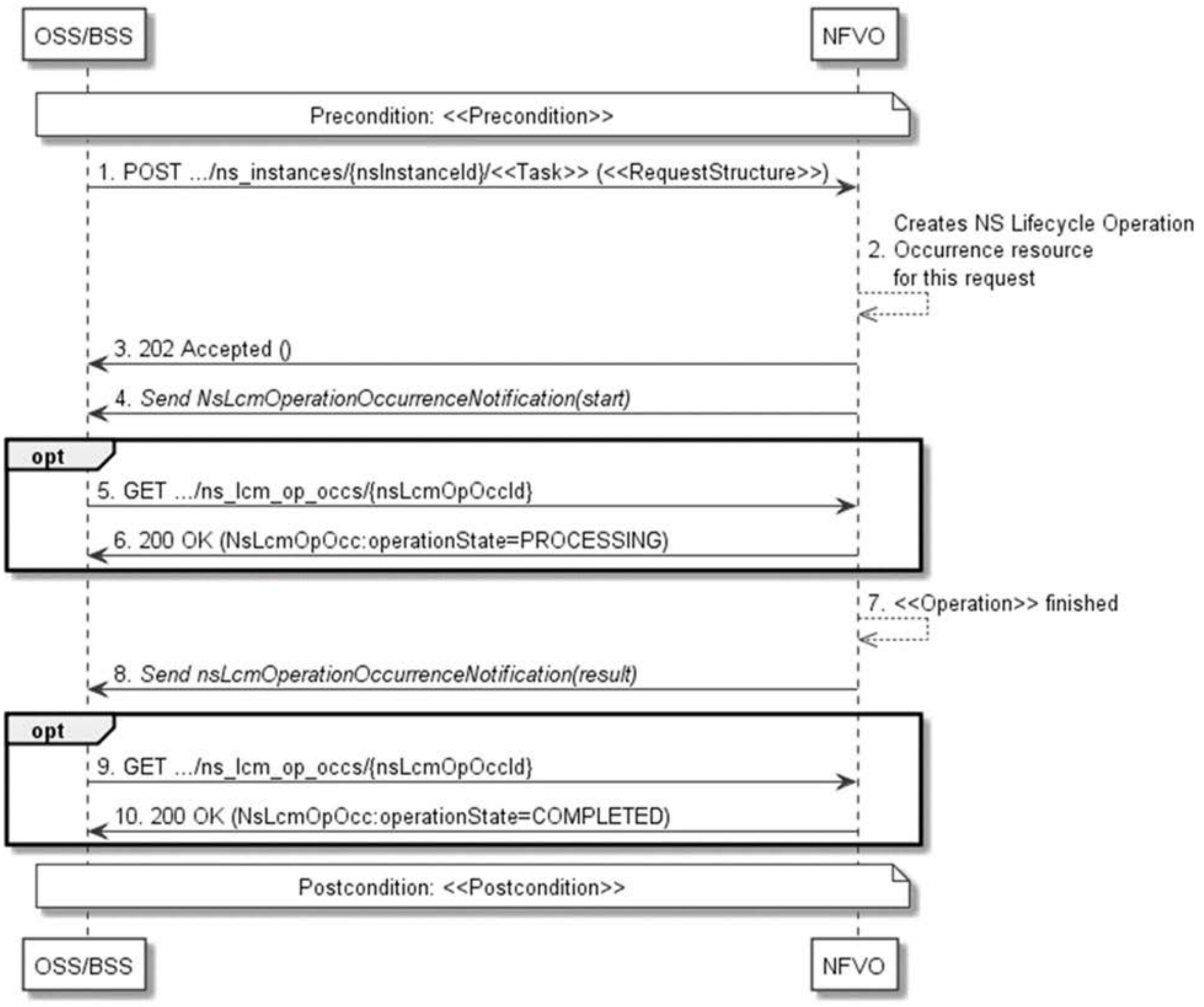
OSS/BSS
NFVO
Precondition: <<Precondition>>
1. POST .../ns_instances/{nsInstanceId}/<<Task>> (<<RequestStructure>>)
2. Creates NS Lifecycle Operation Occurrence resource for this request
3. 202 Accepted ()
4. Send NsLcmOperationOccurrenceNotification(start)
opt
5. GET .../ns_lcm_op_occs/{nsLcmOpOccId}
6. 200 OK (NsLcmOpOcc:operationState=PROCESSING)
7. <<Operation>> finished
8. Send nsLcmOperationOccurrenceNotification(result)
opt
9. GET .../ns_lcm_op_occs/{nsLcmOpOccId}
10. 200 OK (NsLcmOpOcc:operationState=COMPLETED)
Postcondition: <<Postcondition>>
OSS/BSS
NFVO

Figure 6.3.3-1: Flow of NS lifecycle operations triggered by task resources

NOTE: Due to possible race conditions, the 202 response and the "start" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

Precondition: The precondition depends on the actual operation and is described by the template parameter <<Precondition>> in Table 6.3.3-1.

A NS lifecycle operation, as illustrated in Figure 6.3.3-1, consists of the following steps:

1) The OSS/BSS sends a POST request to the <<Task>> resource that represents the lifecycle operation to be executed on the NS instance, and includes in the entity 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 Table 6.3.3-1.

2) The NFVO creates a "NS Lifecycle Operation Occurrence" resource for the request.

3) The NFVO returns a "202 Accepted" response with an empty entity body and a "Location" HTTP header that points to the new "NS Lifecycle Operation Occurrence" resource, i.e. it includes the URI of that resource which is "…/ns_lcm_op_occs/{nsLcmOpOccId}."

4) The NFVO sends to the OSS/BSS a lifecycle management operation occurrence notification (see clause 6.3.6) to indicate the start of the lifecycle management operation occurrence.

5) If desired, the NFVO can poll the "NS Lifecycle Operation Occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the NS Lifecycle Operation Occurrence.

6) In the response to that request, the NFVO returns to the OSS/BSS information of the operation, such as the operation status, by providing in the entity body a data structure of type "NsLcmOpOcc."

7) The NFVO has finished the operation <<Operation>>.

8) The NFVO sends a lifecycle management operation occurrence notification (see clause 6.3.6) to indicate the completion of the lifecycle management operation occurrence.

9) If desired, the OSS/BSS can send a new GET request to the "NS Lifecycle Operation Occurrence" resource.

10) In the response to that request, the NFVO returns to the OSS/BSS information about the result of the operation, by providing in the entity body a data structure of type "NsLcmOpOcc".

Postcondition: The postcondition depends on the actual operation and is described by the template parameter <<Postcondition>> in Table 6.3.3-1.

Error handling: If the NS 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 NS lifecycle management operation occurrence related to this NS lifecycle management operation. Table 6.3.3-1 defines how the flow described above is parameterized for the different NS lifecycle management operations.

Table 6.3.3-1: Parameterization of the flow for different NS lifecycle management operations

Operation	Precondition	Task	RequestStructure	Postcondition
Instantiate NS	NS instance created and in NOT_INSTANTIATED state	instantiate	InstantiateNsRequest	NS instance in INSTANTIATED state
Scale NS	NS instance in INSTANTIATED state	scale	ScaleNsRequest	NS instance still in INSTANTIATED state and NS was scaled
Update NS	NS instance in INSTANTIATED state	update	UpdateNsRequest	NS instance still in INSTANTIATED state and NS was updated
Heal NS	NS instance in INSTANTIATED state	heal	HealNsRequest	NS instance still in INSTANTIATED state
Terminate NS	NS instance in INSTANTIATED state	terminate	TerminateNsRequest	NS instance in NOT_INSTANTIATED state

6.3.4 Flow of the get operations status operation

This clause describes a sequence for obtaining the status of a NS lifecycle management operation occurrence.

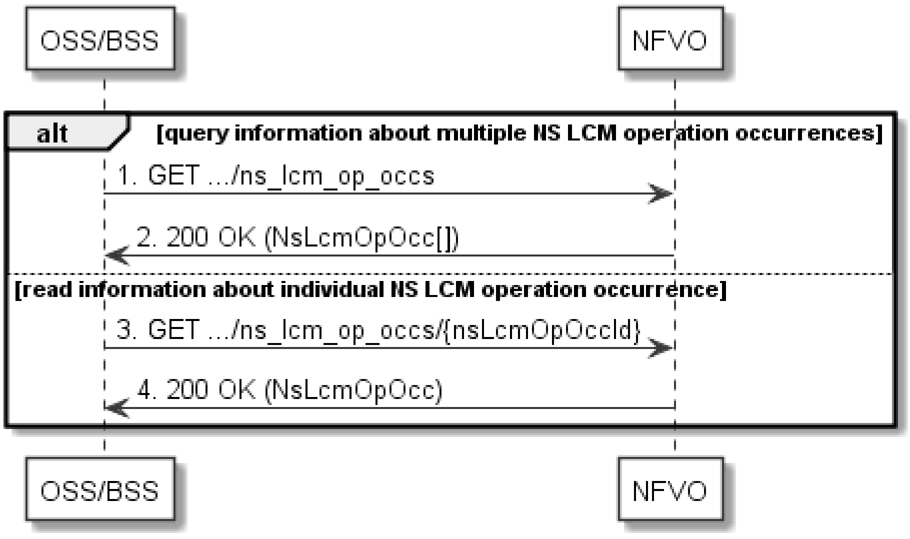
OSS/BSS
NFVO
alt
[query information about multiple NS LCM operation occurrences]
1. GET .../ns_lcm_op_occs
2. 200 OK (NsLcmOpOcc[])
[read information about individual NS LCM operation occurrence]
3. GET .../ns_lcm_op_occs/{nsLcmOpOccId}
4. 200 OK (NsLcmOpOcc)
OSS/BSS
NFVO

Figure 6.3.4-1: Flow of get NS lifecycle operation status

Obtaining the NS lifecycle operation status, as illustrated in Figure 6.3.4-1, consists of the following steps:

1) If the OSS/BSS intends to query all NS lifecycle management operation occurrences, it sends a GET request to the "NS LCM operation occurrences" resource.

2) The NFVO returns a "200 OK" response to the OSS/BSS, and includes zero or more data structures of type "NsLcmOpOcc" in the payload body.

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

4) The NFVO returns a "200 OK" response to the OSS/BSS, and includes one data structure of type "NsLcmOpOcc" in the payload body.

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

6.3.5 Flow of managing subscriptions

This clause describes the procedure for creating, reading, and terminating subscriptions to notifications related to NS lifecycle management.

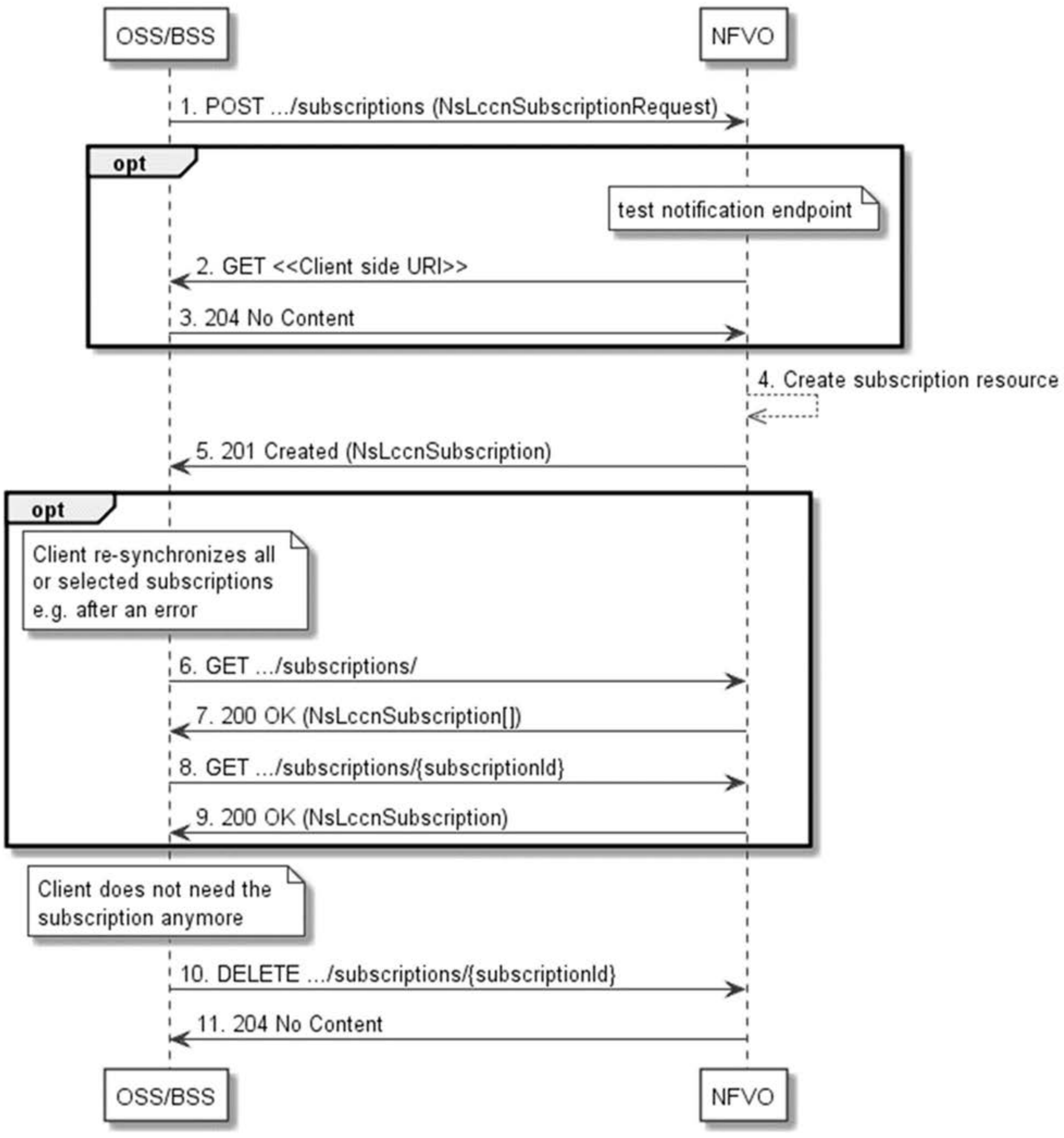
OSS/BSS
NFVO
1. POST .../subscriptions (NsLccnSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (NsLccnSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (NsLccnSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (NsLccnSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
OSS/BSS
NFVO

Figure 6.3.5-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in Figure 6.3.5-1:

1) The OSS/BSS sends a POST request to the "Subscriptions" resource including in the entity body a data structure of type "NsLccnSubscriptionRequest". That data structure contains filtering criteria and a client side URI to which the NFVO will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the OSS/BSS as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

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

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

5) The NFVO returns a "201 Created" response containing a data structure of type "NsLccnSubscription" representing the subscription resource just created by the NFVO, 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 OSS/BSS may obtain information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the OSS/BSS.

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

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

10) If the OSS/BSS 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 OSS/BSS acknowledges the successful termination of the subscription by returning a "204 No Content" response.

6.3.6 Flow of sending notifications

This clause describes the procedure for sending notifications related to NS lifecycle management.

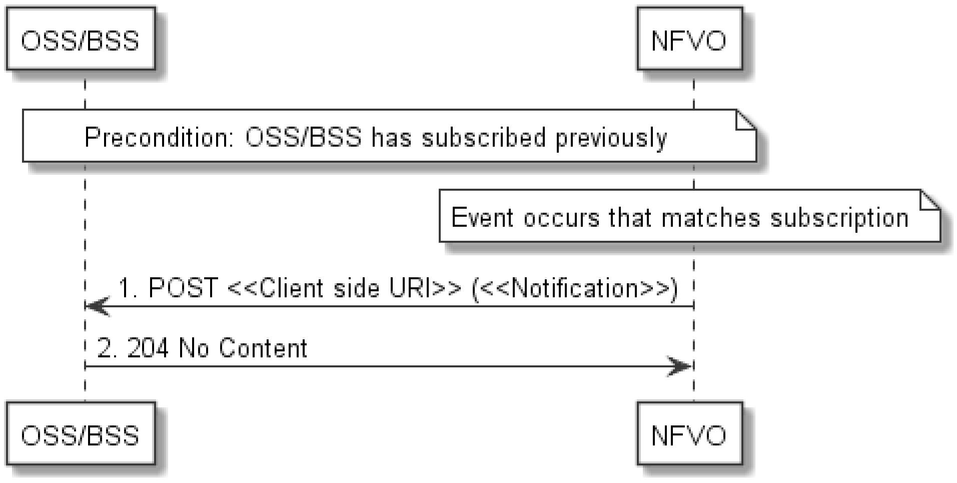
OSS/BSS
NFVO
Precondition: OSS/BSS has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
OSS/BSS
NFVO

Figure 6.3.6-1: Flow of sending notifications

The procedure consists of the following steps as illustrated in Figure 6.3.6-1.

Precondition: The OSS/BSS has subscribed previously to notifications related to NS lifecycle management.

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the OSS/BSS 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 6.5.2.5 through 6.5.2.8).

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

6.3.7 Flow of retrying a NS lifecycle management operation

This clause describes a sequence for retrying a NS lifecycle management operation occurrence that is represented by a "NS 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 NS lifecycle management errors is provided in clause 6.6.

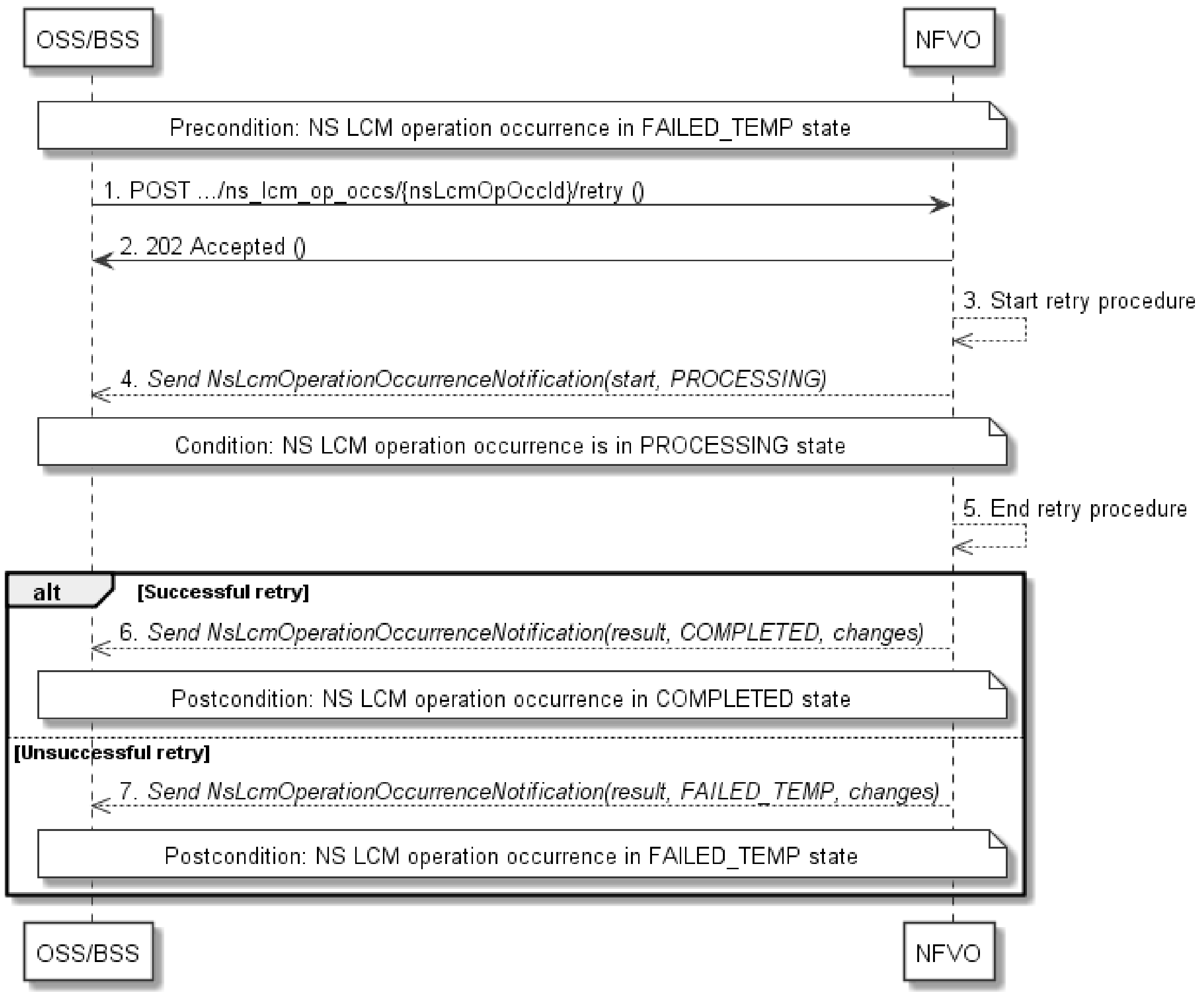
OSS/BSS
NFVO
Precondition: NS LCM operation occurrence in FAILED_TEMP state
1. POST .../ns_lcm_op_occs/{nsLcmOpOccId}/retry ()
2. 202 Accepted ()
3. Start retry procedure
4. Send NsLcmOperationOccurrenceNotification(start, PROCESSING)
Condition: NS LCM operation occurrence is in PROCESSING state
5. End retry procedure
alt
[Successful retry]
6. Send NsLcmOperationOccurrenceNotification(result, COMPLETED, changes)
Postcondition: NS LCM operation occurrence in COMPLETED state
[Unsuccessful retry]
7. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: NS LCM operation occurrence in FAILED_TEMP state
OSS/BSS
NFVO

Figure 6.3.7-1: Flow of retrying a NS lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "PROCESSING" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

Precondition: The NS lifecycle operation occurrence is in FAILED_TEMP state.

Retrying a NS lifecycle operation, as illustrated in Figure 6.3.7-1, consists of the following steps:

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

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

3) The NFVO starts the retry procedure.

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

5) The NFVO finishes the retry procedure.

6) On successful retry, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate successful completion of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).

7) On unsuccessful retry, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate an intermediate error (retry failed) of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).

Postcondition: The NS lifecycle operation occurrence resource is in one of the following states: FAILED_TEMP, COMPLETED. COMPLETED is a terminal state (see clause 6.6.2.2).

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

6.3.8 Flow of rolling back a NS lifecycle management operation

This clause describes a sequence for rolling back a NS lifecycle management operation occurrence that is represented by a "NS 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 NS lifecycle management errors is provided in clause 6.6.

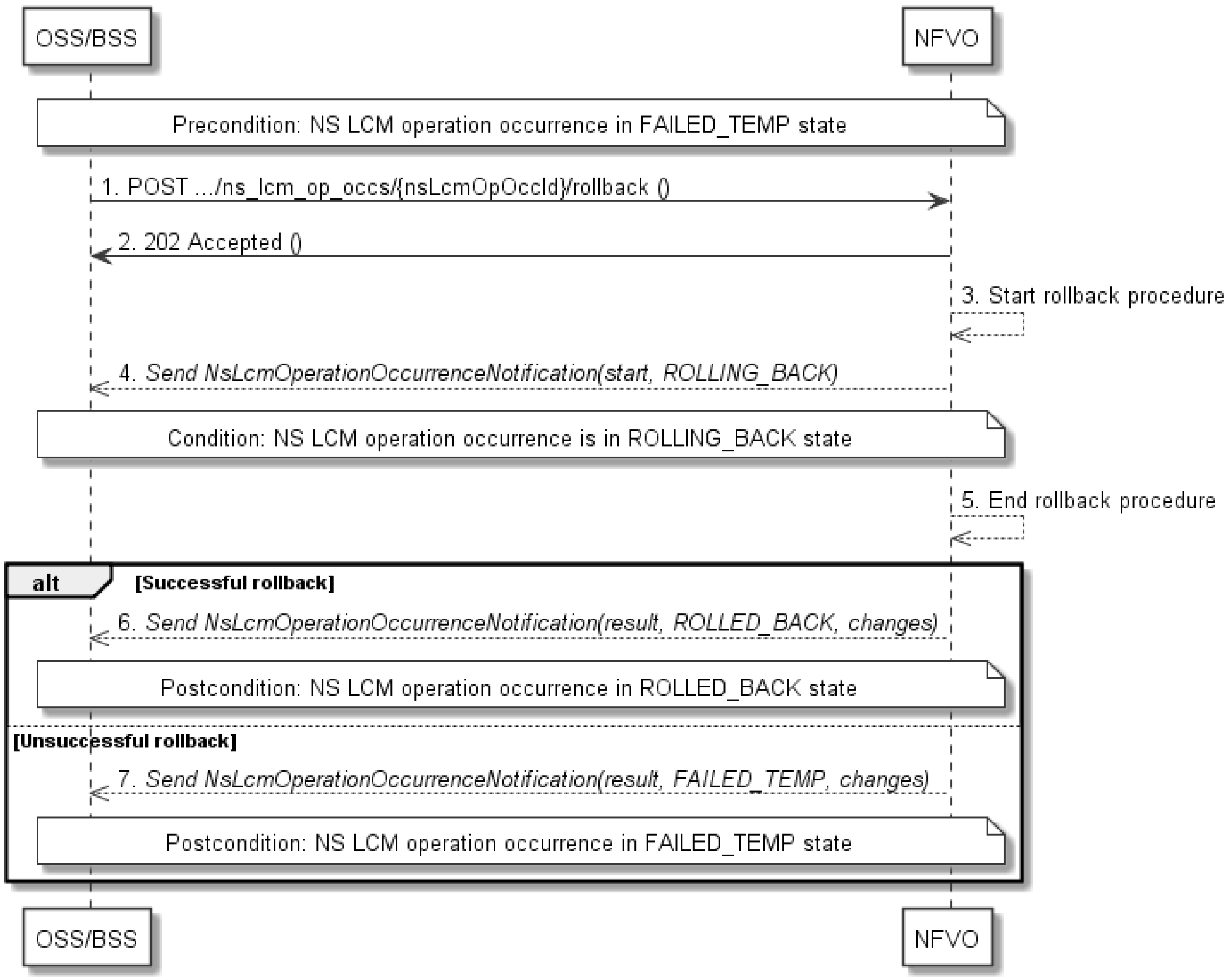
OSS/BSS
NFVO
Precondition: NS LCM operation occurrence in FAILED_TEMP state
1. POST .../ns_lcm_op_occs/{nsLcmOpOccId}/rollback ()
2. 202 Accepted ()
3. Start rollback procedure
4. Send NsLcmOperationOccurrenceNotification(start, ROLLING_BACK)
Condition: NS LCM operation occurrence is in ROLLING_BACK state
5. End rollback procedure
alt
[Successful rollback]
6. Send NsLcmOperationOccurrenceNotification(result, ROLLED_BACK, changes)
Postcondition: NS LCM operation occurrence in ROLLED_BACK state
[Unsuccessful rollback]
7. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: NS LCM operation occurrence in FAILED_TEMP state
OSS/BSS
NFVO

Figure 6.3.8-1: Flow of rolling back a NS lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "ROLLING_BACK" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

Precondition: The NS lifecycle operation occurrence is in FAILED_TEMP state.

Initiating the rollback of a NS lifecycle management operation, as illustrated in Figure 6.3.8-1, consists of the following steps:

1) The OSS/BSS sends a POST request with an empty body to the "Rollback operation task" resource of the NS LCM operation occurrence that is to be rolled back.
2) The NFVO returns a "202 Accepted" response.
3) The NFVO starts the rollback procedure.
4) The NFVO sends a lifecycle management operation occurrence notification of type "start" to indicate that the NS LCM operation occurrence enters the "ROLLING_BACK" state.
5) The NFVO finishes the rollback procedure.
6) On successful rollback, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate successful completion of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).
7) On unsuccessful retry, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate an intermediate error (rollback failed) of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).

Postcondition: The NS lifecycle operation occurrence resource is in one of the following states: FAILED_TEMP, ROLLED_BACK. ROLLED_BACK is a terminal state (see clause 6.6.2.2).

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

6.3.9 Flow of continuing a NS lifecycle management operation

This clause describes a sequence for continuing a NS lifecycle management operation occurrence that is represented by a "NS LCM operation occurrence" resource. Continue is used if an operation is in FAILED_TEMP state, and there is reason to believe that the current operation can continue despite the error. The error can be fixed later, typically after current NS lifecycle management operation finishes.

A comprehensive description of the handling of NS lifecycle management errors is provided in clause 6.6.

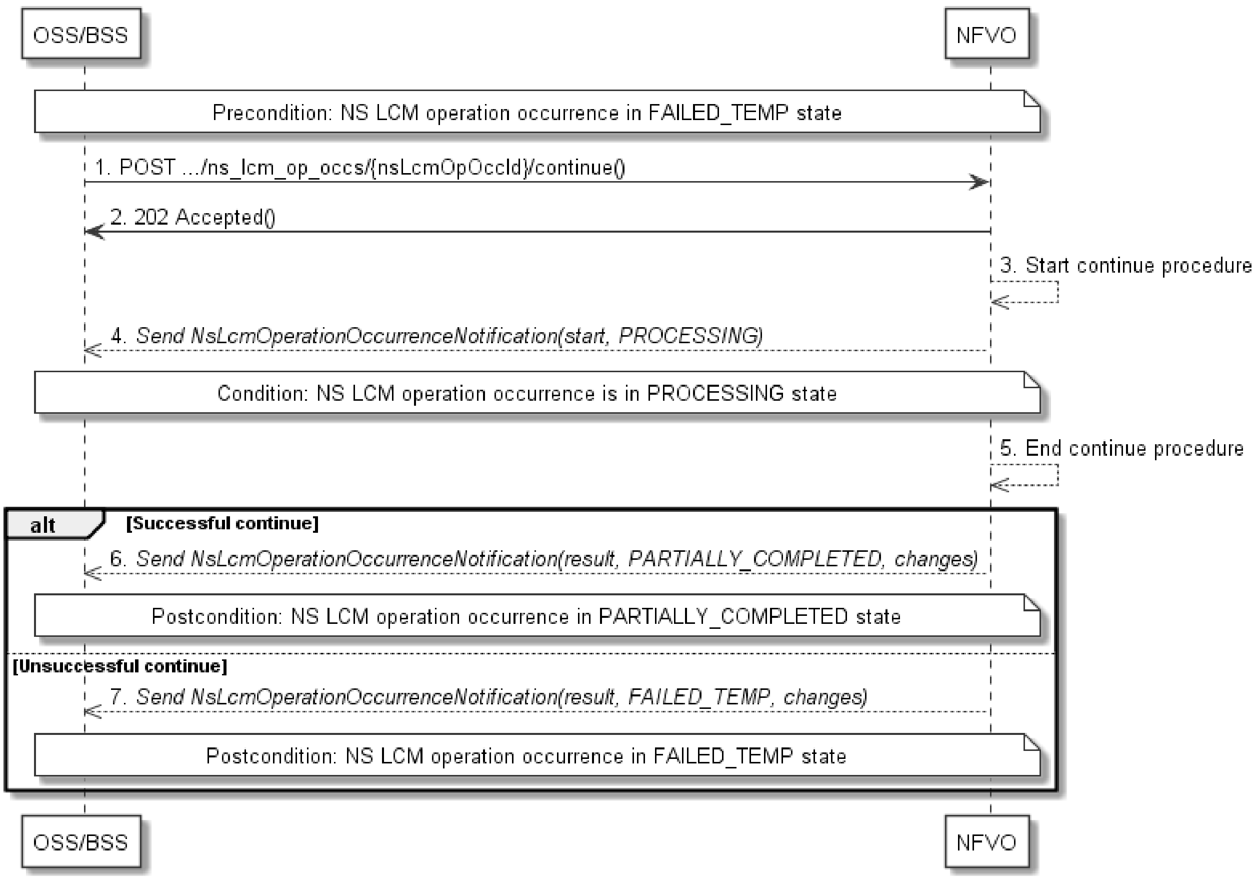
OSS/BSS
NFVO
Precondition: NS LCM operation occurrence in FAILED_TEMP state
1. POST .../ns_lcm_op_occs/{nsLcmOpOccId}/continue()
2. 202 Accepted()
3. Start continue procedure
4. Send NsLcmOperationOccurrenceNotification(start, PROCESSING)
Condition: NS LCM operation occurrence is in PROCESSING state
5. End continue procedure
alt
[Successful continue]
6. Send NsLcmOperationOccurrenceNotification(result, PARTIALLY_COMPLETED, changes)
Postcondition: NS LCM operation occurrence in PARTIALLY_COMPLETED state
[Unsuccessful continue]
7. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: NS LCM operation occurrence in FAILED_TEMP state
OSS/BSS
NFVO

Figure 6.3.9-1: Flow of continuing a NS lifecycle management operation

NOTE: Due to possible race conditions, the 202 response and the "PROCESSING" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

Precondition: The NS lifecycle operation occurrence is in FAILED_TEMP state.

Continuing a NS lifecycle operation, as illustrated in Figure 6.3.9-1, consists of the following steps:

1) The OSS/BSS sends a POST request with an empty body to the "Continue operation task" resource of the NS LCM operation occurrence that is to be retried.

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

3) The NFVO starts the continue procedure.

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

5) The NFVO finishes the continue procedure.

6) On successful continue, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate successful completion of the operation, and inform the OSS/BSS about the resources changes.

7) On unsuccessful continue, the NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate an intermediate error (continue failed) of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).

Postcondition: The NS lifecycle operation occurrence resource is in one of the following states: FAILED_TEMP, PARTIALLY_COMPLETED. PARTIALLY_COMPLETED is a terminal state (see clause 6.6.2.2).

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

6.3.10 Flow of failing a NS lifecycle management operation

This clause describes a sequence for declaring as "failed" a NS lifecycle management operation occurrence that is represented by a "NS 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 HealNs, or non-graceful NS termination, on the affected NS instance.

A comprehensive description of the handling of NS lifecycle management errors is provided in clause 6.6.

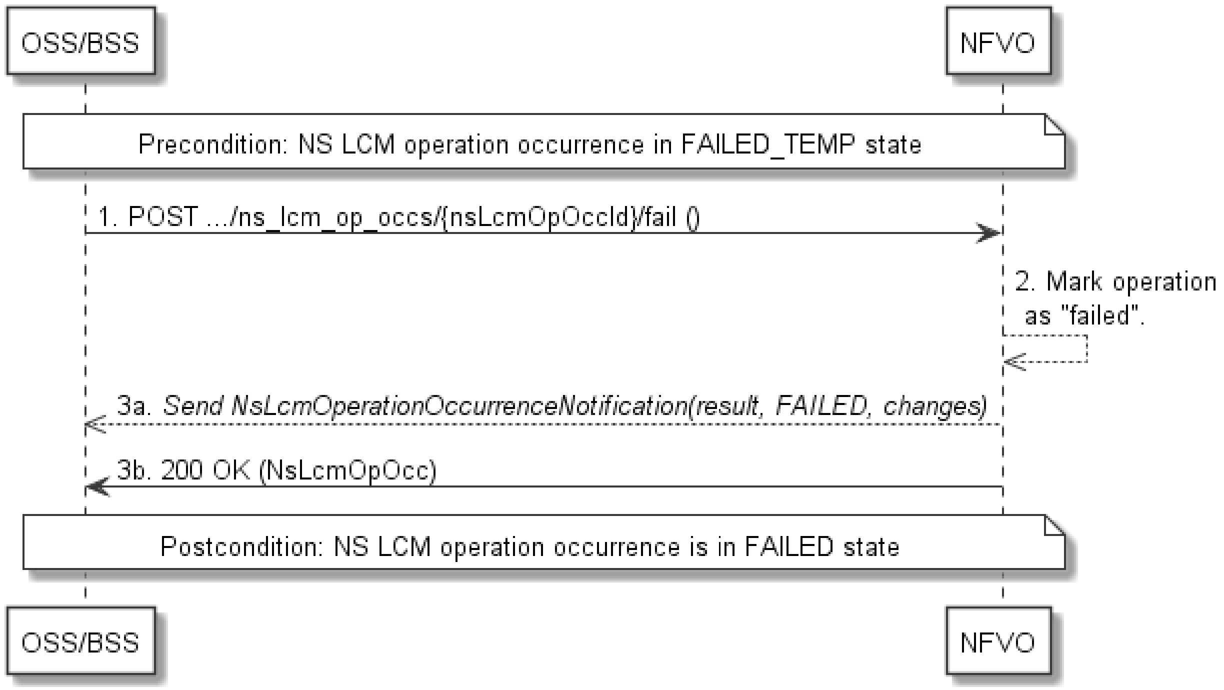
OSS/BSS
NFVO
Precondition: NS LCM operation occurrence in FAILED_TEMP state
1. POST .../ns_lcm_op_occs/{nsLcmOpOccId}/fail ()
2. Mark operation as "failed".
3a. Send NsLcmOperationOccurrenceNotification(result, FAILED, changes)
3b. 200 OK (NsLcmOpOcc)
Postcondition: NS LCM operation occurrence is in FAILED state
OSS/BSS
NFVO

Figure 6.3.10-1: Flow of declaring a NS lifecycle management operation as failed

NOTE: Due to possible race conditions, the 200 response and the "FAILED" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

Precondition: The NS lifecycle operation occurrence is in FAILED_TEMP state.

Declaring a NS lifecycle management operation as failed, as illustrated in Figure 6.3.10-1, consists of the following steps:

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

2) The NFVO marks the operation as failed.

3) The NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate the final failure of the operation, and inform the OSS/BSS about the s changes on the NS components (e.g. VNFs, VLs). Furthermore, it returns a "200 OK" response, and includes in the body a NsLcmOpOcc structure. The order in which the response and the notification arrive at the OSS/BSS is not defined.

Postcondition: The NS lifecycle operation occurrence resource is FAILED state. This is a terminal state (see clause 6.6.2.2).

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

6.3.11 Flow of cancelling a NS lifecycle management operation

This clause describes a sequence for cancelling an ongoing NS LCM operation occurrence, or a rollback of a NS 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 NS lifecycle management errors is provided in clause 6.6.

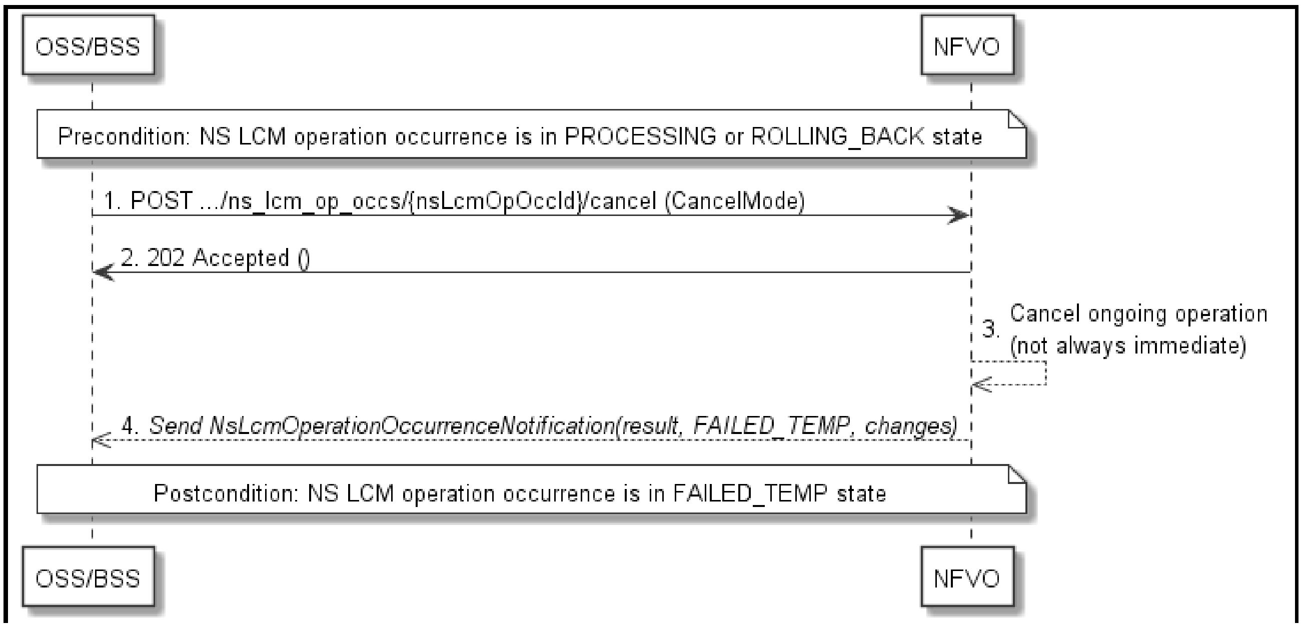
OSS/BSS
NFVO
Precondition: NS LCM operation occurrence is in PROCESSING or ROLLING_BACK state
1. POST .../ns_lcm_op_occs/{nsLcmOpOccId}/cancel (CancelMode)
2. 202 Accepted ()
3. Cancel ongoing operation (not always immediate)
4. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Postcondition: NS LCM operation occurrence is in FAILED_TEMP state
OSS/BSS
NFVO

Figure 6.3.11-1: Flow of cancelling a NS lifecycle management operation in "PROCESSING" or "ROLLING_BACK" state

NOTE: Due to possible race conditions, the 202 response and the "FAILED_TEMP" NsLcmOperationOccurrenceNotification can arrive in any order at the OSS/BSS.

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

Cancelling a NS lifecycle operation when it is in "PROCESSING" or "ROLLING_BACK" state, as illustrated in Figure 6.3.11-1, consists of the following steps:

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

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

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

4) The NFVO sends a NS lifecycle management operation occurrence notification (see clause 6.3.6) to indicate an intermediate error (cancelled) of the operation, and inform the OSS/BSS about the changes on the NS components (e.g. VNFs, VLs).

Postcondition: The NS lifecycle management operation occurrence resource is FAILED_TEMP state.

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

6.4 Resources

6.4.1 Introduction

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

6.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the NS lifecycle management interface.

6.4.2 Resource: NS Instances

6.4.2.1 Description

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

6.4.2.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances

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 new NS instance resource.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.2.3.1-1 and 6.4.2.3.1-2.

Table 6.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.2.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality		Remarks
	CreateNsRequest	1	The Ns creation parameters, as defined in clause 6.5.2.7.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	NsInstance	1	201 Created	A NS Instance identifier was created successfully. The response body shall contain a representation of the created NS instance, as defined in clause 6.5.2.8. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the created NS 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.

6.4.2.3.2 GET

The GET method queries information about multiple NS instances.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.2.3.2-1 and 6.4.2.3.2-2.

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 expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the NsInstance and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO 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 NFVO shall support this parameter. The following attributes shall be excluded from the NsInstance structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - vnfInstances - pnfInfo - virtualLinkInfo - vnffgInfo - sapInfo - nsScaleStatus - additionalAffinityOrAntiAffinityRules
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 6.4.2.3.2-2: Details of the GET request/response on this resource

Request body	Data type	Cardinality		Remarks
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	NsInstance	0..N	200 OK	Information about zero or more NS instances was queried successfully. The response body shall contain in an array the representations of zero or more NS instances, as defined in clause 6.5.2.8. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

6.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3 Resource: Individual NS Instance

6.4.3.1 Description

This resource represents an individual NS instance. The client can use this resource to modify, delete, and query the underlying NS instance.

6.4.3.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}

The base resource URI variables for this resource are 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
nsInstanceId	Identifier of the NS instance

6.4.3.3 Resource methods

6.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.2 GET

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

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.3.3.2-1 and 6.4.3.3.2-2.

Table 6.4.3.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.3.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Remarks	
Request body	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	NsInstance	1	200 OK	Information about an individual NS instance was queried successfully. The response body shall contain a representation of the NS instance, as defined in clause 6.5.2.8.
	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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.3.3.5 DELETE

This method deletes an individual NS instance resource.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.3.3.5-1 and 6.4.3.3.5-2.

Table 6.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.3.3.5-2: Details of the DELETE request/response on this resource

Request body	**Data type**	**Cardinality**	**Remarks**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	n/a		204 No Content	The NS instance resource and the associated NS 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 NS 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.

6.4.4 Resource: Instantiate NS task

6.4.4.1 Description

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

6.4.4.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}/instantiate

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.
nsInstanceId	Identifier of the NS instance to be instantiated.

6.4.4.3 Resource methods

6.4.4.3.1 POST

The POST method requests to instantiate a NS instance resource.

This method shall follow the provisions specified in the Tables 6.4.4.3.1-1 and 6.4.4.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
None supported		

Table 6.4.4.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	InstantiateNsRequest	1	Parameters for the instantiate NS operation, as defined in clause 6.5.2.10.	
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 "NS 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 NS instance resource is in the 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.

6.4.4.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5 Resource: Scale NS task

6.4.5.1 Description

This task resource represents the "Scale NS" operation. The client can use this resource to request to scale a NS instance. Scaling an NS instance can be performed by explicitly adding/removing existing VNF instances to/from the NS instance, by leveraging on the abstraction mechanism provided by the NS scaling aspects and NS levels information elements declared in the NSD or by scaling individual VNF instances that are part of the NS itself. When adding VNFs and nested NSs - already existing or not - to the NS to be scaled, the NFVO shall follow the indications provided by the dependencies attribute, as specified in the corresponding NSD.

NOTE: In case the NS is a composite NS, it is also possible to scale directly its nested NS, as they are also NS and thus indirectly effectively scale the composite NS.

6.4.5.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}/scale

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.
nsInstanceId	Identifier of the NS instance to be scaled.

6.4.5.3 Resource methods

6.4.5.3.1 POST

The POST method requests to scale a NS instance resource.

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

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**	**Description**	
	ScaleNsRequest	1	Parameters for the scale NS operation, as defined in clause 6.5.2.13.	
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 "NS lifecycle 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 NS 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.

6.4.5.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6 Resource: Update NS task

6.4.6.1 Description

This task resource represents the "Update NS" operation. The client can use this resource to update a NS instance.

6.4.6.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}/update

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.
nsInstanceId	Identifier of the NS instance to be updated.

6.4.6.3 Resource methods

6.4.6.3.1 POST

The POST method requests to update a NS instance resource.

This method shall follow the provisions specified in the Tables 6.4.6.3.1-1 and 6.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.6.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.6.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**		**Description**
	UpdateNsRequest	1		Parameters for the update NS operation, as defined in clause 6.5.2.11.
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 "NS lifecycle 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 NS 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.

6.4.6.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7 Resource: Heal NS task

6.4.7.1 Description

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

6.4.7.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}/heal

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.
nsInstanceId	Identifier of the NS instance to be healed.

6.4.7.3 Resource methods

6.4.7.3.1 POST

The POST method requests to heal a NS instance resource.

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.

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

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**		**Description**
	HealNsRequest	1		Parameters for the heal NS operation, as defined in clause 6.5.2.12.
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 "NS lifecycle 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 NS 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.

6.4.7.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8 Resource: Terminate NS task

6.4.8.1 Description

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

6.4.8.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_instances/{nsInstanceId}/terminate

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.
nsInstanceId	The identifier of the NS instance to be terminated.

6.4.8.3 Resource methods

6.4.8.3.1 POST

The POST method terminates a NS instance. This method can only be used with a NS instance in the INSTANTIATED state. Terminating a NS instance does not delete the NS instance identifier, but rather transitions the NS into the NOT_INSTANTIATED state.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.8.3.1-1 and 6.4.8.3.1-2.

Table 6.4.8.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.8.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Remarks
Request body	TerminateNsRequest	1		The terminate NS request parameters, as defined in clause 6.5.2.14.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	n/a	1	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 "NS lifecycle 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 NS 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.

6.4.8.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.8.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9 Resource: NS LCM operation occurrences

6.4.9.1 Description

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

6.4.9.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs

The base resource URI variables for this resource are defined in Table 6.4.9.2-1.

Table 6.4.9.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.9.3 Resource methods

6.4.9.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9.3.2 GET

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

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
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the NsLcmOpOcc and in data types referenced from it shall be supported by the NFVO in the filter expression.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO 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 NFVO shall support this parameter. The following attributes shall be excluded from the NsLcmOpOcc structure in the response body if this parameter is provided: - operationParams - changedVnfInfo - error - resourceChanges
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 6.4.9.3.2-2: 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**
	NsLcmOpOcc	0..N	200 OK	Status information for zero or more NS lifecycle management operation occurrences was queried successfully. The response body shall contain in an array the representations of zero or more NS instances, as defined in clause 5.5.2.13. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

6.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.9.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.10 Resource: Individual NS LCM operation occurrence

6.4.10.1 Description

This resource represents a NS lifecycle management operation occurrence. The client can use this resource to read information about a NS 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.

6.4.10.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}

The base resource URI variables for this resource are defined in Table 6.4.10.2-1.

Table 6.4.10.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
nsLcmOpOccId	Identifier of a NS lifecycle management operation occurrence

6.4.10.3 Resource methods

6.4.10.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.10.3.2 GET

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

This method shall follow the provisions specified in the Tables 6.4.10.3.2-1 and 6.4.10.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.10.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.10.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	NsLcmOpOcc	1	200 OK	Information about a NS LCM operation occurrence was queried successfully. The response body shall contain status information about a NS lifecycle management operation occurrence (see 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.

6.4.10.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.10.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.10.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.11 Resource: Retry operation task

6.4.11.1 Description

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

6.4.11.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}/retry

This resource shall support the resource URI variables defined in Table 6.4.11.2-1.

Table 6.4.11.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
nsLcmOpOccId	Identifier of a NS 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 POST request triggering a NS LCM operation. It can also be retrieved from the "nsLcmOpOccId" attribute in the NsLcmOperationOccurrenceNotification.	

6.4.11.3 Resource methods

6.4.11.3.1 POST

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

This method shall follow the provisions specified in the Tables 6.4.11.3.1-1 and 6.4.11.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.11.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.11.3.1-2: 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 reason can also be that the task is not supported for the NS LCM operation occurrence represented by the parent resource, and 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 NS LCM operation occurrence resource. Typically, this is due to the fact that the NS 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.

6.4.11.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.11.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.11.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.11.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.12 Resource: Rollback operation task

6.4.12.1 Description

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

6.4.12.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}/rollback

This resource shall support the resource URI variables defined in Table 6.4.12.2-1.

Table 6.4.12.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
nsLcmOpOccId	Identifier of a NS 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 POST request triggering a NS LCM operation. It can also be retrieved from the "nsLcmOpOccId" attribute in the NsLcmOperationOccurrenceNotification.	

6.4.12.3 Resource methods

6.4.12.3.1 POST

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

This method shall follow the provisions specified in the Tables 6.4.12.3.1-1 and 6.4.12.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.12.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.12.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	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 reason can also be that the task is not supported for the NS LCM operation occurrence represented by the parent resource, and 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 NS LCM operation occurrence resource. Typically, this is due to the fact that the NS 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.

6.4.12.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.12.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.12.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.12.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.13 Resource: Continue operation task

6.4.13.1 Description

This task resource represents the "Continue operation" operation. The client can use this resource to initiate continuing an NS lifecycle management operation.

6.4.13.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}/continue

This resource shall support the resource URI variables defined in Table 6.4.13.2-1.

Table 6.4.13.2-1: Resource URI variables for this resource

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

6.4.13.3 Resource methods

6.4.13.3.1 POST

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

This method shall follow the provisions specified in the Tables 6.4.13.3.1-1 and 6.4.13.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.13.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.13.3.1-2: 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 reason can also be that the task is not supported for the NS LCM operation occurrence represented by the parent resource, and 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 NS LCM operation occurrence resource. Typically, this is due to the fact that the NS 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.

6.4.13.3.2 GET

Not supported.

6.4.13.3.3 PUT

Not supported.

6.4.13.3.4 PATCH

Not supported.

6.4.13.3.5 DELETE

Not supported.

6.4.14 Resource: Fail operation task

6.4.14.1 Description

This task resource represents the "Fail operation" operation. The client can use this resource to mark a NS lifecycle management operation occurrence as "finally failed", i.e. change the state of the related NS 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.

6.4.14.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}/fail

This resource shall support the resource URI variables defined in Table 6.4.14.2-1.

Table 6.4.14.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
nsLcmOpOccId	Identifier of a NS 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 POST request triggering a NS LCM operation. It can also be retrieved from the "nsLcmOpOccId" attribute in the NsLcmOperationOccurrenceNotification.	

6.4.14.3 Resource methods

6.4.14.3.1 POST

The POST method marks a NS 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 6.4.14.3.1-1 and 6.4.14.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.14.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
None supported		

Table 6.4.14.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
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**
	NsLcmOpOcc	1	200 OK	The state of the NS lifecycle management operation occurrence was changed successfully. The response shall include a representation of the NS lifecycle management 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 reason can also be that the task is not supported for the NS LCM operation occurrence represented by the parent resource, and 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 NS LCM operation occurrence resource. Typically, this is due to the fact that the NS 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.

6.4.14.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.14.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.14.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.14.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.15 Resource: Cancel operation task

6.4.15.1 Description

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

6.4.15.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}/cancel

This resource shall support the resource URI variables defined in Table 6.4.15.2-1.

Table 6.4.15.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
nsLcmOpOccId	Identifier of a NS 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 POST request triggering a NS LCM operation. It can also be retrieved from the "nsLcmOpOccId" attribute in the NsLcmOperationOccurrenceNotification.	

6.4.15.3 Resource methods

6.4.15.3.1 POST

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

This method shall follow the provisions specified in the Tables 6.4.15.3.1-1 and 6.4.15.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.15.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
None supported		

Table 6.4.15.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	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.
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 entity 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 reason can also be that the task is not supported for the NS LCM operation occurrence represented by the parent resource, and 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 NS 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.

6.4.15.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.15.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.15.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.15.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.16 Resource: Subscriptions

6.4.16.1 Description

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

6.4.16.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/subscriptions

This resource shall support the resource URI variables defined in Table 6.4.16.2-1.

Table 6.4.16.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2

6.4.16.3 Resource methods

6.4.16.3.1 POST

The POST method creates a new subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.16.3.1-1 and 6.4.16.3.1-2.

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 OSS, and might make sense only in very rare use cases. Consequently, the NFVO 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 6.4.16.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.16.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Remarks	
Request body	LccnSubscriptionRequest	1	Details of the subscription to be created, as defined in clause 6.5.2.2.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	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 exits and the policy of the NFVO 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.

6.4.16.3.2 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 support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.16.3.2-1 and 6.4.16.3.2-2.

Table 6.4.16.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the LccnSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 6.4.16.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Remarks
Request body	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	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. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

6.4.16.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.16.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.16.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.17 Resource: Individual subscription

6.4.17.1 Description

This resource represents an individual subscription. It can be used by the client to read and to terminate a subscription to Notifications related to NS lifecycle management.

6.4.17.2 Resource definition

The resource URI is:

{apiRoot}/nslcm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in Table 6.4.17.2-1.

Table 6.4.17.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2
subscriptionId	Identifier of this subscription

6.4.17.3 Resource methods

6.4.17.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.17.3.2 GET

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

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.17.3.2-1 and 6.4.17.3.2-2.

Table 6.4.17.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.17.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Remarks
Request body	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	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.

6.4.17.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.17.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.17.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.17.3.5-1 and 6.4.17.3.5-2.

Table 6.4.17.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Remarks
n/a		

Table 6.4.17.3.5-2: Details of the DELETE request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	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.

6.4.18 Resource: Notification endpoint

6.4.18.1 Description

This resource represents a notification endpoint. The server can use this resource to send notifications to a subscribed client, which has provided the URI of this resource during the subscription process.

6.4.18.2 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 6.4.18.2-1.

Table 6.4.18.2-1: Resource URI variables for this resource

Name	Definition
n/a	

6.4.18.3 Resource methods

6.4.18.3.1 POST

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

This method shall support the URI query parameters, request and response data structures, and response codes, as specified in the Tables 6.4.18.3.1-1 and 6.4.18.3.1-2.

Table 6.4.18.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
n/a		

Each notification request body shall include exactly one of the alternatives defined in Table 6.4.18.3.1-2.

Table 6.4.18.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Remarks	
Request body	NsLcmOperationOccurrenceNotification	1	A notification about lifecycle changes triggered by a NS LCM operation occurrence.	
	NsIdentifierCreationNotification	1	A notification about the creation of a NS identifier and the related NS instance resource.	
	NsIdentifierDeletionNotification	1	A notification about the deletion of a NS identifier and the related NS instance resource.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Remarks**
	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.

6.4.18.3.2 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 6.4.18.3.2-1 and 6.4.18.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 6.4.18.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 6.4.18.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	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.

6.4.18.3.3 PUT

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.18.3.4 PATCH

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.4.18.3.5 DELETE

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

6.5 Data model

6.5.1 Introduction

This clause defines the request and response data structures of the NS Lifecycle management interface.

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: LccnSubscriptionRequest

This type represents a subscription request related to notifications about NS lifecycle changes. It shall comply with the provisions defined in Table 6.5.2.2-1.

Table 6.5.2.2-1: 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.

6.5.2.3 Type: NsLcmOpOcc

This type represents a request a NS lifecycle operation occurrence. It shall comply with the provisions defined in Table 6.5.2.3-1.

Table 6.5.2.3-1: Definition of the NsLcmOpOcc data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this NS lifecycle operation occurrence.
operationState	NsLcmOperationStateType	1	The state of the NS LCM operation.
statusEnteredTime	DateTime	1	Date-time when the current state was entered.
nsInstanceId	Identifier	1	Identifier of the NS instance to which the operation applies.
lcmOperationType	NsLcmOpType	1	Type of the actual LCM operation represented by this lcm operation occurrence.
startTime	DateTime	1	Date-time of the start of the operation.
isAutomaticInvocation	Boolean	1	Set to true if this NS LCM operation occurrence has been automatically triggered by the NFVO. This occurs in the case of auto-scaling, auto-healing and when a nested NS is modified as a result of an operation on its composite NS. Set to false otherwise.

Attribute name	Data type	Cardinality	Description
operationParams	Object	0..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 lcmOperationType and the data type of this attribute shall apply: • INSTANTIATE: InstantiateNsRequest • SCALE: ScaleNsRequest • UPDATE: UpdateNsRequest • HEAL: HealNsRequest • TERMINATE: TerminateNsRequest This attribute shall be present if this data type is returned in a response to reading an individual resource, and may be present according to the chosen attribute selector parameter if this data type is returned in a response to a query of a container resource.
isCancelPending	Boolean	1	If the LCM operation occurrence is in "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 "operationState" 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.
>affectedVnfs	AffectedVnf	0..N	Information about the VNF instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
>affectedPnfs	AffectedPnf	0..N	Information about the PNF instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
>affectedVls	AffectedVl	0..N	Information about the VL instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
>affectedVnffgs	AffectedVnffg	0..N	Information about the VNFFG instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
>affectedNss	AffectedNs	0..N	Information about the nested NS instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
>affectedSaps	AffectedSap	0..N	Information about the SAP instances that were affected during the lifecycle operation, if this notification represents the result of a lifecycle operation. See note.
_links	Structure (inline)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>nsInstance	Link	1	Link to the NS instance that the operation applies to.
>cancel	Link	0..1	Link to the task resource that represents the "cancel" operation for this LCM operation occurrence, if cancelling is currently allowed.
>retry	Link	0..1	Link to the task resource that represents the "retry" operation for this LCM operation occurrence, if retrying is currently allowed.
>rollback	Link	0..1	Link to the task resource that represents the "rollback" operation for this LCM operation occurrence, if rolling back is currently allowed.
>continue	Link	0..1	Link to the task resource that represents the "continue" operation for this LCM operation occurrence, if continuing is currently allowed.

Attribute name	Data type	Cardinality	Description
>fail	Link	0..1	Link to the task resource that represents the "fail" operation for this LCM operation occurrence, if declaring as failed is currently allowed.
NOTE: This allows the OSS/BSS to obtain a copy of the latest "result" notification if it has not received it due to an error. If the notification represents the successful result of a lifecycle operation, at least an affectedVnf, or affectedPnf, or affectedVl, or affectedVnffg or affectedNs, or affectedSap shall be present.			

6.5.2.4 Type: LccnSubscription

This type represents a subscription related to notifications about NS lifecycle changes. It shall comply with the provisions defined in Table 6.5.2.4-1.

Table 6.5.2.4-1: 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.

6.5.2.5 Type: NsLcmOperationOccurrenceNotification

This type represents an NS lifecycle management operation occurrence notification, which informs the receiver of changes in the NS lifecycle caused by an NS LCM operation occurrence. The NS LCM operation occurrence may be triggered by the OSS/BSS or automatically triggered by the NFVO. The automatic trigger occurs in case of auto-scaling, auto-healing and when a nested NS is modified as a result of an operation on its composite NS.

It shall comply with the provisions defined in Table 6.5.2.5-1. The support of the notification is mandatory.

This notification shall be triggered by the NFVO when there is a change in the NS lifecycle caused by an LCM operation occurrence, including:

- Instantiation of the NS (start and result)
- Scaling of the NS (start and result, including the auto-scaling)
- Update of the NS (start and result)
- Termination of the NS (start and result)
- Healing of the NS (start and result, including the auto-healing)
- When a nested NS is modified as a result of an operation on its composite NS

If this is the initial notification about the start of an LCM operation occurrence, the notification shall be sent by the NFVO before any action is taken as part of , the LCM operation. Due to possible race conditions, the "start" notification and the LCM operation acknowledgment can arrive in any order at the OSS/BSS, and the OSS/BSS shall be able to handle such a situation.

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

Table 6.5.2.5-1: Definition of the NsLcmOperationOccurrenceNotification 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.
nsInstanceId	Identifier	1	The identifier of the NS instance affected.
nsLcmOpOccId	Identifier	1	The identifier of the NS lifecycle operation occurrence associated to the notification.
operation	LcmOpType	1	The lifecycle operation.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "NsLcmOperationOccurrenceNotification" 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 NS lifecycle operation or the result of a NS lifecycle operation. Permitted values: - START: Informs about the start of the NS LCM operation occurrence. - RESULT: Informs about the final or intermediate result of the NS LCM operation occurrence.
operationState	NsLcmOperationStateType	1	The state of the NS lifecycle operation occurrence.
isAutomaticInvocation	Boolean	1	Set to true if this NS LCM operation occurrence has been automatically triggered by the NFVO. This occurs in case of auto-scaling, auto-healing and when a nested NS is modified as a result of an operation on its composite NS. Set to false otherwise.
affectedVnf	AffectedVnf	0..N	Information about the VNF instances that were affected during the lifecycle operation. See note.
affectedPnf	AffectedPnf	0..N	Information about the PNF instances that were affected during the lifecycle operation. See note.
affectedVl	AffectedVirtualLink	0..N	Information about the VL instances that were affected during the lifecycle operation. See note.
affectedVnffg	AffectedVnffg	0..N	Information about the VNFFG instances that were affected during the lifecycle operation. See note.
affectedNs	AffectedNs	0..N	Information about the NS instances that were affected during the lifecycle operation. See note.
affectedSap	AffectedSap	0..N	Information about the SAP instances that were affected during the lifecycle operation. See note.
error	ProblemDetails	0..1	Details of the latest error, if one has occurred during executing the LCM operation (see clause 4.3.5). Shall be present if operationState 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.			

6.5.2.6 Type: NsIdentifierCreationNotification

This type represents a NS identifier creation notification, which informs the receiver of the creation of a new NS instance resource and the associated NS instance identifier. It shall comply with the provisions defined in Table 6.5.2.6-1. The support of the notification is mandatory. This notification shall be triggered by the NFVO when it has created a NS instance resource and the associated NS instance identifier.

Table 6.5.2.6-1: Definition of the NsIdentifierCreationNotification data type

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Discriminator for the different notification types. Shall be set to "NsIdentifierCreationNotification" 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.
nsInstanceId	Identifier	1	The created NS instance identifier.
_links	LccnLinks	1	Links to resources related to this notification.

6.5.2.7 Type: NsIdentifierDeletionNotification

This type represents a NS identifier deletion notification, which informs the receiver of the deletion of a new NS instance resource and the associated NS instance identifier. It shall comply with the provisions defined in Table 6.5.2.7-1. The support of the notification is mandatory. This notification shall be triggered by the NFVO when it has deleted a NS instance resource and the associated NS instance identifier.

Table 6.5.2.7-1: Definition of the NsInstanceDeletionNotification data type

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Discriminator for the different notification types. Shall be set to "NsIdentifierDeletionNotification" 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.
nsInstanceId	Identifier	1	The deleted NS instance identifier.
_links	LccnLinks	1	Links to resources related to this notification.

6.5.2.8 Type: NsChangeNotification

This type represents a NS change notification, which informs the receiver of changes on the NS instance caused by the LCM operation occurrence, which directly or indirectly impacts its NS component and is triggered without any context of this NS instance. This notification is different from the NsLcmOperationOccurenceNotification (see clause 6.5.2.5), which is triggered by the LCM operation occurrence on the NS instance itself. It shall comply with the provisions defined in Table 6.5.2.8-1. The support of the notification is mandatory.

The trigger conditions include:

- LCM operation occurrence which directly or indirectly impacts the NS component (start and result)

If this is a notification about the start of an LCM operation occurrence impacting the NS component, the notification shall be provided as soon as the impact on the NS component is identified.

If this is a notification about a final result state of an LCM operation occurrence impacting the NS component, the notification shall be provided after the impact on the NS component has been executed.

Table 6.5.2.8-1: Definition of the NsChangeNotification data type

Attribute name	Data type	Cardinality	Description
nsInstanceId	Identifier	1	The identifier of the NS instance affected.
nsComponentType	NsComponentType	1	Indicates the impacted NS component type.
nsComponentId	Identifier	1	The identifier of the impacted NS component.
lcmOpOccIdImpactingNsComponent	Identifier	1	The identifier of the lifecycle operation occurrence which is associated to the notification and impacts the NS component directly or indirectly.
lcmOpNameImpactingNsComponent	LcmOpNameForChangeNotificationType	1	Indicates the name of the lifecycle operation occurrence which is associated to the notification and impacts the NS component directly or indirectly.

Attribute name	Data type	Cardinality	Description
lcmOpOccStatusImpactingNsComponent	LcmOpOccStatusForChangeNotificationType	1	Indicates this status of the lifecycle operation occurrence which is associated to the notification and impacts the NS component directly or indirectly.
notificationType	String	1	Discriminator for the different notification types. Shall be set to "NsChangeNotification" 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.
error	ProblemDetails	0..1	Details of the latest error, if one has occurred during executing the LCM operation (see clause 4.3.5). Shall be present if lcmOpOccStatusImpactingNsComponent is "PARTIALLY_COMPLETED" or "FAILED", and shall be absent otherwise.
_links	LccnLinks	1	Links to resources related to this notification.

6.5.2.9 Type: CreateNsRequest

This type represents a request for the NS identifier creation operation. It shall comply with the provisions defined in Table 6.5.2.9-1.

Table 6.5.2.9-1: Definition of the CreateNsRequest data type

Attribute name	Data type	Cardinality	Description
nsdId	Identifier	1	Identifier of the NSD that defines the NS instance to be created.
nsName	String	1	Human-readable name of the NS instance to be created.
nsDescription	String	1	Human-readable description of the NS instance to be created.

6.5.2.10 Type: NsInstance

This type represents a response for Query NS operation. It shall comply with the provisions defined in Table 6.5.2.10-1.

Table 6.5.2.10-1: Definition of the NsInstance data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the NS instance.
nsInstanceName	String	1	Human readable name of the NS instance.
nsInstanceDescription	String	1	Human readable description of the NS instance.
nsdId	Identifier	1	Identifier of the NSD on which the NS instance is based.
nsdInfoId	Identifier	1	Identifier of the NSD information object on which the NS instance is based. This identifier was allocated by the NFVO.
flavourId	IdentifierInNsd	0..1	Identifier of the NS deployment flavour applied to the NS instance. This attribute shall be present if the nsState attribute value is INSTANTIATED.
vnfInstance	VnfInstance	0..N	Information on constituent VNF(s) of the NS instance. See note.
pnfInfo	PnfInfo	0..N	Information on the PNF(s) that are part of the NS instance.
virtualLinkInfo	NsVirtualLinkInfo	0..N	Information on the VL(s) of the NS instance. This attribute shall be present if the nsState attribute value is INSTANTIATED and if the NS instance has specified connectivity.
vnffgInfo	VnffgInfo	0..N	Information on the VNFFG(s) of the NS instance.
sapInfo	SapInfo	0..N	Information on the SAP(s) of the NS instance.
nestedNsInstanceId	Identifier	0..N	Identifier of the nested NS(s) of the NS instance. See note.

Attribute name	Data type	Cardinality	Description
nsState	Enum (inlined)	1	The state of the NS instance. Permitted values: NOT_INSTANTIATED: The NS instance is terminated or not instantiated. INSTANTIATED: The NS instance is instantiated.
monitoringParameter	NsMonitoringParameter	0..N	Performance metrics tracked by the NFVO (e.g. for auto-scaling purposes) as identified by the NS designer in the NSD.
nsScaleStatus	NsScaleInfo	0..N	Status of each NS scaling aspect declared in the applicable DF, how "big" the NS instance has been scaled w.r.t. that aspect. This attribute shall be present if the nsState attribute value is INSTANTIATED.
additionalAffinityOrAntiAffinityRule	AffinityOrAntiAffinityRule	0..N	Information on the additional affinity or anti-affinity rule from NS instantiation operation. Shall not conflict with rules already specified in the NSD.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>nestedNsInstances	Link	0..N	Links to the nested NS instances of the present NS instance.
>instantiate	Link	0..1	Link to the "instantiate" task resource, if the related operation is possible based on the current status of this NS instance resource (i.e. NS 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 NS instance resource (i.e. NS instance is in INSTANTIATED state).
>update	Link	0..1	Link to the "update" task resource, if the related operation is possible based on the current status of this NS instance resource (i.e. NS instance is in INSTANTIATED state).
>scale	Link	0..1	Link to the "scale" task resource, if the related operation is supported for this NS instance, and is possible based on the current status of this NS instance resource (i.e. NS instance is in INSTANTIATED state).
>heal	Link	0..1	Link to the "heal" task resource, if the related operation is supported for this NS instance, and is possible based on the current status of this NS instance resource (i.e. NS instance is in INSTANTIATED state).
NOTE: If the "nsState" attribute is INSTANTIATED, at least either one "vnfInstance" attribute or one "nestedNsInstanceId" attribute shall be present.			

6.5.2.11 Type: InstantiateNsRequest

This operation supports the instantiation of a NS instance. It shall comply with the provisions defined in Table 6.5.2.11-1.

Table 6.5.2.11-1: Definition of the InstantiateNsRequest data type

Attribute name	Data type	Cardinality	Description
nsFlavourId	IdentifierInNsd	1	Identifier of the NS deployment flavour to be instantiated.
sapData	SapData	0..N	Create data concerning the SAPs of this NS.
addpnfData	AddPnfData	0..N	Information on the PNF(s) that are part of this NS.
vnfInstanceData	VnfInstanceData	0..N	Specify an existing VNF instance to be used in the NS. If needed, the VNF Profile to be used for this VNF instance is also provided. See note 1.

Attribute name	Data type	Cardinality	Description
nestedNsInstanceData	NestedNsInstanceDatat	0..N	Specify an existing NS instance to be used as a nested NS within the NS. If needed, the NS Profile to be used for this nested NS instance is also provided. See notes 2 and 3.
localizationLanguage	VnfLocationConstraint	0..N	Defines the location constraints for the VNF to be instantiated as part of the NS instantiation. An example can be a constraint for the VNF to be in a specific geographic location.
additionalParamsForNs	KeyValuePairs	0..1	Allows the OSS/BSS to provide additional parameter(s) at the composite NS level (as opposed to the VNF level, which is covered in additionalParamsForVnf), and as opposed to the nested NS level, which is covered in additionalParamForNestedNs.
additionalParamForNestedNs	ParamsForNestedNs	0..N	Allows the OSS/BSS to provide additional parameter(s) per nested NS instance (as opposed to the composite NS level, which is covered in additionalParamForNs, and as opposed to the VNF level, which is covered in additionalParamForVnf). This is for nested NS instances that are to be created by the NFVO as part of the NS instantiation and not for existing nested NS instances that are referenced for reuse.
additionalParamsForVnf	ParamsForVnf	0..N	Allows the OSS/BSS to provide additional parameter(s) per VNF instance (as opposed to the composite NS level, which is covered in additionalParamsForNs), and as opposed to the nested NS level, which is covered in additionalParamForNestedNs). This is for VNFs that are to be created by the NFVO as part of the NS instantiation and not for existing VNF that are referenced for reuse.
startTime	DateTime	0..1	Timestamp indicating the earliest time to instantiate the NS. Cardinality "0" indicates the NS instantiation takes place immediately.
nsInstantiationLevelId	IdentifierInNsd	0..1	Identifies one of the NS instantiation levels declared in the DF applicable to this NS instance. If not present, the default NS instantiation level as declared in the NSD shall be used.
additionalAffinityOrAntiAffinityRule	AffinityOrAntiAffinityRule	0..N	Specifies additional affinity or anti-affinity constraint for the VNF instances to be instantiated as part of the NS instantiation. Shall not conflict with rules already specified in the NSD.
NOTE 1: The DF of the VNF instance shall match the VNF DF present in the associated VNF Profile. NOTE 2: The NS DF of each nested NS shall be one of the allowed flavours in the associated NSD (as referenced in the nestedNsd attribute of the NSD of the NS to be instantiated). NOTE 3: The NSD of each referenced NSs (i.e. each nestedInstanceId) shall match the one of the nested NSD in the composite NSD.			

6.5.2.12 Type: UpdateNsRequest

This operation supports the update of a NS instance. It shall comply with the provisions defined in Table 6.5.2.12-1.

Table 6.5.2.12-1: Definition of the UpdateNsRequest data type

Attribute name	Data type	Cardinality	Description
updateType	Enum (inlined)	1	The type of update. It determines also which one of the following parameters is present in the operation. Possible values include: - ADD_VNF: Adding existing VNF instance(s) - REMOVE_VNF: Removing VNF instance(s) - INSTANTIATE_VNF: Instantiating new VNF(s) - CHANGE_VNF_DF: Changing VNF DF - OPERATE_VNF: Changing VNF state - MODIFY_VNF_INFORMATION: Modifying VNF information and/or the configurable properties of VNF instance(s)

Attribute name	Data type	Cardinality	Description
			- CHANGE_EXTERNAL_VNF_CONNECTIVITY: Changing the external connectivity of VNF instance(s)ADD_SAP: Adding SAP(s) - REMOVE_SAP: Removing SAP(s) - ADD_NESTED_NS: Adding existing NS instance(s) as nested NS(s) - REMOVE_NESTED_NS: Removing existing nested NS instance(s) - ASSOC_NEW_NSD_VERSION: Associating a new NSD version to the NS instance - MOVE_VNF: Moving VNF instance(s) from one origin NS instance to another target NS instance - ADD_VNFFG: Adding VNFFG(s) - REMOVE_VNFFG: Removing VNFFG(s) - UPDATE_VNFFG: Updating VNFFG(s) - CHANGE_NS_DF: Changing NS DF - ADD_PNF: Adding PNF - MODIFY_PNF: Modifying PNF - REMOVE_PNF: Removing PNF
addVnfIstance	VnfInstanceData	0..N	Identifies an existing VNF instance to be added to the NS instance. It shall be present only if updateType = "ADD_VNF".
removeVnfInstanceId	Identifier	0..N	Identifies an existing VNF instance to be removed from the NS instance. It contains the identifier(s) of the VNF instances to be removed. It shall be present only if updateType = "REMOVE_VNF." Note: If a VNF instance is removed from a NS and this NS was the last one for which this VNF instance was a part, the VNF instance is terminated by the NFVO.
instantiateVnfData	InstantiateVnfData	0..N	Identifies the new VNF to be instantiated. It can be used e.g. for the bottom-up NS creation. It shall be present only if updateType = "INSTANTIATE_VNF".
changeVnfFlavourData	ChangeVnfFlavourData	0..N	Identifies the new DF of the VNF instance to be changed to. It shall be present only if updateType = "CHANGE_VNF_DF".
operateVnfData	OperateVnfData	0..N	Identifies the state of the VNF instance to be changed. It shall be present only if updateType = "OPERATE_VNF".
modifyVnfInfoData	ModifyVnfInfoData	0..N	Identifies the VNF information parameters and/or the configurable properties of VNF instance to be modified. It shall be present only if updateType = "MODIFY_VNF_INFORMATION".
changeExtVnfConnectivityData	ChangeExtVnfConnectivityData	0..N	Specifies the new external connectivity data of the VNF instance to be changed. It shall be present only if updateType = "CHANGE_EXTERNAL_VNF_CONNECTIVITY".
addSap	SapData	0..N	Identifies a new SAP to be added to the NS instance. It shall be present only if updateType = "ADD_SAP."
removeSapId	Identifier	0..N	The identifier an existing SAP to be removed from the NS instance. It shall be present only if updateType = "REMOVE_SAP."
addNestedNsData	NestedNsInstanceData	0..N	The identifier of an existing nested NS instance to be added to (nested within) the NS instance. It shall be present only if updateType = "ADD_NESTED_NS".
removeNestedNsId	IdentiferInNs	0..N	The identifier of an existing nested NS instance to be removed from the NS instance. It shall be present only if updateType = "REMOVE_NESTED_NS".
assocNewNsdVersionData	AssocNewNsdVersionData	0..1	Specify the new NSD to be used for the NS instance. It shall be present only if updateType = ASSOC_NEW_NSD_VERSION".
moveVnfInstanceData	MoveVnfInstanceData	0..N	Specify existing VNF instance to be moved from one NS instance to another NS instance. It shall be present only if updateType = MOVE_VNF".
addVnffg	AddVnffgData	0..N	Specify the new VNFFG to be created to the NS Instance. It shall be present only if updateType = "ADD_VNFFG".

Attribute name	Data type	Cardinality	Description
removeVnffgId	Identifier	0..N	Identifier of an existing VNFFG to be removed from the NS Instance. It shall be present only if updateType = "REMOVE_VNFFG".
updateVnffg	UpdateVnffgData	0..N	Specify the new VNFFG Information data to be updated for a VNFFG of the NS Instance. It shall be present only if updateType = "UPDATE_VNFFG".
changeNsFlavourData	ChangeNsFlavourData	0..1	Specifies the new DF to be applied to the NS instance. It shall be present only if updateType = "CHANGE_NS_DF".
addPnfData	AddPnfData	0..N	Specifies the PNF to be added into the NS instance. It shall be present only if updateType = "ADD_PNF".
modifyPnfData	ModifyPnfData	0..N	Specifies the PNF to be modified in the NS instance. It shall be present only if updateType = "MODIFY_PNF".
removePnfId	Identifier	0..N	Identifier of the PNF to be deleted from the NS instance. It shall be present only if updateType = "REMOVE_PNF".
updateTime	DateTime	0..1	Timestamp indicating the update time of the NS, i.e. the NS will be updated at this timestamp. Cardinality "0" indicates the NS update takes place immediately.

6.5.2.13 Type: HealNsRequest

This operation supports the healing of an NS instance, either by healing the complete NS instance or by healing one of more of the VNF instances that are part of this NS. It shall comply with the provisions defined in Table 6.5.2.13-1.

Table 6.5.2.13-1: Definition of the HealNsRequest data type

Attribute name	Data type	Cardinality	Description
healNsData	HealNsData	0..1	Provides the information needed to heal an NS. See note.
healVnfData	HealVnfData	0..N	Provides the information needed to heal a VNF. See note.
NOTE: Either the parameter healNsData or the parameter healVnfData, but not both shall be provided.			

6.5.2.14 Type: ScaleNsRequest

This type represents a request for the scale NS operation. It shall comply with the provisions defined in Table 6.5.2.14-1.

Table 6.5.2.14-1: Definition of the ScaleNsRequest data type

Attribute name	Data type	Cardinality	Description
scaleType	Enum (inlined)	1	Indicates the type of scaling to be performed. Possible values: - SCALE_NS - SCALE_VNF
scaleNsData	ScaleNsData	0..1	The necessary information to scale the referenced NS instance. It shall be present when scaleType = SCALE_NS. See note.
scaleVnfData	ScaleVnfData	0..N	The necessary information to scale the referenced NS instance. It shall be present when scaleType = SCALE_VNF. See note.
scaleTime	DateTime	0..1	Timestamp indicating the scale time of the NS, i.e. the NS will be scaled at this timestamp. Cardinality "0" indicates the NS scaling takes place immediately.
NOTE: Either the parameter scaleNsData or the parameter scaleVnfData, but not both shall be provided.			

6.5.2.15 Type: TerminateNsRequest

This type represents a NS termination request. It shall comply with the provisions defined in Table 6.5.2.15-1.

Table 6.5.2.15-1: Definition of the TerminateNsRequest data type

Attribute name	Data type	Cardinality	Description
terminationTime	DateTime	0..1	Timestamp indicating the end time of the NS, i.e. the NS will be terminated automatically at this timestamp. Cardinality "0" indicates the NS termination takes place immediately.

6.5.2.16 Type: CancelMode

This type represents a parameter to select the mode of cancelling an ongoing NS LCM operation occurrence. It shall comply with the provisions defined in Table 6.5.2.16-1.

Table 6.5.2.16-1: Definition of the CancelMode data type

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

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: AffectedVnf

This type provides information about added, deleted and modified VNFs. It shall comply with the provisions in Table 6.5.3.2-1.

Table 6.5.3.2-1: Definition of the AffectedVnf data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance.
vnfdId	Identifier	1	Identifier of the VNFD of the VNF Instance.
vnfProfileId	IdentifierInNsd	1	Identifier of the VNF profile of the NSD.
vnfName	String	1	Name of the VNF Instance.
changeType	Enum (inline)	1	Signals the type of change Permitted values: - ADD - REMOVE - INSTANTIATE - TERMINATE - SCALE - CHANGE_FLAVOUR - HEAL - OPERATE - MODIFY_INFORMATION - CHANGE_EXTERNAL_VNF_CONNECTIVITY

Attribute name	Data type	Cardinality	Description
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED
changedInfo	Structure (inline)	0..1	Information about the changed VNF instance information, including VNF configurable properties, if applicable.
>changedVnfInfo	ModifyVnfInfoData	0..1	Information about the changed VNF instance information, including configurable properties, if applicable. See note.
>changedExtConnectivity	ExtVirtualLinkInfo	0..N	Information about changed external connectivity, if applicable. See note.
NOTE: When the "changedInfo" attribute is present, either the "changedVnfInfo" attribute or the "changedExtConnectivity" attribute or both shall be present.			

6.5.3.3 Type: AffectedPnf

This type provides information about added, deleted and modified PNFs. It shall comply with the provisions in Table 6.5.3.3-1.

Table 6.5.3.3-1: Definition of the AffectedPnf data type

Attribute name	Data type	Cardinality	Description
pnfId	Identifier	1	Identifier of the affected PNF. This identifier is allocated by the OSS/BSS.
pnfdId	IdentifierInNsd	1	Identifier of the PNFD on which the PNF is based.
pnfProfileId	IdentifierInNsd	1	Identifier of the PNF profile of the NSD.
pnfName	String	1	Name of the PNF.
cpInstanceId	IdentifierInPnf	1..N	Identifier of the CP in the scope of the PNF.
changeType	Enum (inline)	1	Signals the type of change. Permitted values: - ADD - REMOVE - MODIFY
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED

6.5.3.4 Type: AffectedVirtualLink

This type provides information about added, deleted and modified VLs. It shall comply with the provisions in Table 6.5.3.4-1.

Table 6.5.3.4-1: Definition of the AffectedVirtualLink data type

Attribute name	Data type	Cardinality	Description
nsVirtualLinkInstanceId	IdentifierInNs	1	Identifier of the VL Instance.
nsVirtualLinkDescId	IdentifierInNsd	1	Identifier of the VLD in the NSD for this VL.
vlProfileId	IdentifierInNsd	1	Name of the VL profile.

Attribute name	Data type	Cardinality	Description
changeType	Enum (inline)	1	Signals the type of change. Permitted values: - ADD - DELETE - MODIFY - ADD_LINK_PORT - REMOVE_LINK_PORT
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED

6.5.3.5 Type: AffectedVnffg

This type provides information about added, deleted and modified VNFFG instances. It shall comply with the provisions in Table 6.5.3.5-1.

Table 6.5.3.5-1: Definition of the AffectedVnffg data type

Attribute name	Data type	Cardinality	Description
vnffgInstanceId	IdentifierInNs	1	Identifier of the VNFFG instance.
vnffgdId	IdentifierInNsd	1	Identifier of the VNFFGD of the VNFFG instance.
changeType	Enum (inline)	1	Signals the type of lifecycle change. Permitted values: - ADD - REMOVE - MODIFY
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED

6.5.3.6 Type: AffectedNs

This type provides information about added, deleted and modified nested NSs. It shall comply with the provisions in Table 6.5.3.6-1.

Table 6.5.3.6-1: Definition of the AffectedNs data type

Attribute name	Data type	Cardinality	Description
nsInstanceId	Identifier	1	Identifier of the nested NS instance.
nsdId	Identifier	1	Identifier of the NSD of the nested NS instance.
changeType	Enum (inline)	1	Signals the type of lifecycle change. Permitted values: - ADD - REMOVE - INSTANTIATE - INSTANTIATE - SCALE - UPDATE - HEAL - TERMINATE
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED - PARTIALLY_COMPLETED

6.5.3.7 Type: AffectedSap

This type provides information about added, deleted and modified SAP of a NS. It shall comply with the provisions in Table 6.5.3.7-1.

Table 6.5.3.7-1: Definition of the AffectedSap data type

Attribute name	Data type	Cardinality	Description
sapInstanceId	IdentifierInNs	1	Identifier of the SAP instance.
sapdId	IdentifierInNsd	1	Identifier of the SAPD for this SAP.
sapName	String	1	Human readable name for the SAP.
changeType	Enum (inline)	1	Signals the type of lifecycle change. Permitted values: - ADD - REMOVE - MODIFY
changeResult	Enum (inline)	1	Signals the result of change identified by the "changeType" attribute. Permitted values: - COMPLETED - ROLLED_BACK - FAILED

6.5.3.8 Type: LifecycleChangeNotificationsFilter

This type represents a subscription filter related to notifications about NS lifecycle changes. It shall comply with the provisions defined in Table 6.5.3.8-1.

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 6.5.3.8-1: Definition of the LifecycleChangeNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
nsInstanceSubscriptionFilter	NsInstanceSubscriptionFilter	0..1	Filter criteria to select NS instances about which to notify.
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: - NsLcmOperationOccurenceNotification - NsIdentifierCreationNotification - NsIdentifierDeletionNotification - NsChangeNotification See note.
operationTypes	NsLcmOpType	0..N	Match particular NS lifecycle operation types for the notification of type NsLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "NsLcmOperationOccurrenceNotification", and shall be absent otherwise.
operationStates	LcmOperationStateType	0..N	Match particular LCM operation state values as reported in notifications of type NsLcmOperationOccurrenceNotification. May be present if the "notificationTypes" attribute contains the value "NsLcmOperationOccurrenceNotification", and shall be absent otherwise.
nsComponentTypes	NsComponentType	0..N	Match particular NS component types for the notification of type NsChangeNotification. May be present if the "notificationTypes" attribute contains the value "NsChangeNotification", and shall be absent otherwise.
lcmOpNameImpactingNsComponent	LcmOpNameForChangeNotificationType	0..N	Match particular LCM operation names for the notification of type NsChangeNotification. May be present if the "notificationTypes" attribute contains the value "NsChangeNotification", and shall be absent otherwise.
lcmOpOccStatusImpactingNsComponent	LcmOpOccStatusForChangeNotificationType	0..N	Match particular LCM operation status values as reported in notifications of type NsChangeNotification. May be present if the "notificationTypes" attribute contains the value "NsChangeNotification", 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.			

6.5.3.9 Type: LccnLinks

This type represents the links to resources that a notification can contain. It shall comply with the provisions defined in Table 6.5.3.9-1.

Table 6.5.3.9-1: Definition of the LccnLinks data type

Attribute name	Data type	Cardinality	Description
nsInstance	NotificationLink	1	Link to the resource representing the NS instance to which the notified change applies.
subscription	NotificationLink	1	Link to the subscription that triggered this notification.
nslcmOpOcc	NotificationLink	0..1	Link to the lifecycle operation occurrence that this notification is related to. Shall be present if there is a related lifecycle operation occurrence.

6.5.3.10 Type: SapData

This type represents the information related to a SAP of a NS. It shall comply with the provisions defined in Table 6.5.3.10-1.

Table 6.5.3.10-1: Definition of the SapData data type

Attribute name	Data type	Cardinality	Description
sapdId	IdentifierInNsd	1	Reference to the SAPD for this SAP.
sapName	String	1	Human readable name for the SAP.
description	String	1	Human readable description for the SAP.
sapProtocolData	CpProtocolData	0..N	Parameters for configuring the network protocols on the SAP.

6.5.3.11 Type: CpProtocolData

This type represents network protocol data. It shall comply with the provisions defined in Table 6.5.3.11-1.

Table 6.5.3.11-1: 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.			

6.5.3.12 Type: IpOverEthernetAddressData

This type represents network address data for IP over Ethernet. It shall comply with the provisions defined in Table 6.5.3.12-1.

Table 6.5.3.12-1: 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 NFV MANO. See note 1.
ipAddresses	Structure (inlined)	0..N	List of IP addresses to assign to the extCP 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.			

6.5.3.13 Type: PnfInfo

This type represents the information about a PNF that is part of an NS instance. It shall comply with the provisions defined in Table 6.5.3.13-1.

Table 6.5.3.13-1: Definition of the PnfInfo data type

Attribute name	Data type	Cardinality	Description
pnfId	Identifier	1	Identifier of the PNF. This identifier is allocated by the OSS/BSS.
pnfName	String	1	Name of the PNF.
pnfdId	Identifier	1	Identifier of the PNFD on which the PNF is based.
pnfdInfoId	Identifier	1	Identifier of the PNFD information object related to this PNF. This identifier is allocated by the NFVO.
pnfProfileId	IdentifierInNsd	1	Identifier of the related PnfProfile in the NSD on which the PNF is based.
cpInfo	PnfExtCpInfo	1..N	Information on the external CP of the PNF.

6.5.3.14 Type: AddPnfData

This type specifies an PNF to be added to the NS instance and the PNF Profile to use for this PNF. It shall comply with the provisions defined in Table 6.5.3.14-1.

Table 6.5.3.14-1: Definition of the AddPnfData data type

Attribute name	Data type	Cardinality	Description
pnfId	Identifier	1	Identifier of the PNF. This identifier is allocated by the OSS/BSS.
pnfName	String	1	Name of the PNF.
pnfdId	Identifier	1	Identifier of the PNFD on which the PNF is based.
pnfProfileId	IdentifierInNsd	1	Identifier of related PnfProfile in the NSD on which the PNF is based.
cpData	PnfExtCpData	0..N	Address assigned for the PNF external CP(s).

6.5.3.15 Type: ModifyPnfData

This type specifies an PNF to be modified in the NS instance. It shall comply with the provisions defined in Table 6.5.3.15-1.

Table 6.5.3.15-1: Definition of the ModifyPnfData data type

Attribute name	Data type	Cardinality	Description
pnfId	Identifier	1	Identifier of the PNF. This identifier is allocated by the OSS/BSS.
pnfName	String	0..1	Name of the PNF. See note.
cpData	PnfExtCpData	0..N	Address assigned for the PNF external CP(s). See note.
NOTE: At least one attribute shall be present.			

6.5.3.16 Type: PnfExtCpData

This type represents the configuration data on the external CP of the PNF. It shall comply with the provisions defined in Table 6.5.3.16-1.

Table 6.5.3.16-1: Definition of the PnfExtCpData data type

Attribute name	Data type	Cardinality	Description
cpInstanceI16	IdentifierInPnf	0..1	Identifier of the CP. Shall be present for existing CP.
cpdId	IdentifierInNsd	0..1	Identifier of the Connection Point Descriptor (CPD) for this CP. Shall be present for new CP.
cpProtocolData	CpProtocolData	1..N	Address assigned for this CP.

6.5.3.17 Type: PnfExtCpInfo

This type represents the information about the external CP of the PNF. It shall comply with the provisions defined in Table 6.5.3.17-1.

Table 6.5.3.17-1: Definition of the PnfExtCpInfo data type

Attribute name	Data type	Cardinality	Description
cpInstanceId	IdentifierInPnf	1	Identifier of the CP in the scope of the PNF.
cpdId	IdentifierInNsd	1	Identifier of (reference to) the Connection Point Descriptor (CPD) for this CP.
cpProtocolData	cpProtocolData	1..N	Parameters for configuring the network protocols on the CP.

6.5.3.18 Type: IpOverEthernetAddressInfo

This type represents information about a network address that has been assigned. It shall comply with the provisions defined in Table 6.5.3.18-1.

Table 6.5.3.18-1: Definition of the IpOverEthernetAddressInfo data type

Attribute name	Data type	Cardinality	Description
macAddress	MacAddress	0..1	Assigned MAC address.
ipAddresses	Structure (inlined)	0..N	Addresses assigned to the CP or SAP 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.			

6.5.3.19 Type: VnfInstanceData

This type specifies an existing VNF instance to be used in the NS instance and if needed, the VNF Profile to use for this VNF instance. It shall comply with the provisions defined in Table 6.5.3.19-1.

Table 6.5.3.19-1: Definition of the VnfInstanceData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the existing VNF instance to be used in the NS.
vnfProfileId	IdentifierInNsd	0..1	Identifier of (Reference to) a vnfProfile defined in the NSD which the existing VNF instance shall be matched with. If not present, the NFVO will select the VnfProfile matching the information in the VNF instance.

6.5.3.19a Type: NestedNsInstanceData

This type specifies an existing nested NS instance to be used in the NS instance and if needed, the NsProfile to use for this nested NS instance. It shall comply with the provisions defined in Table 6.5.3.19a-1.

Table 6.5.3.19a-1: Definition of the NestedNsInstanceData data type

Attribute name	Data type	Cardinality	Description
nestedNsInstanceId	Identifier	1	Identifier of the existing nested NS instance to be used in the NS.
nsProfileId	IdentifierInNsd	0..1	Identifier of an NsProfile defined in the NSD which the existing nested NS instance shall be matched with. If not present, the NFVO will select the NsProfile matching the information in the nested NS instance.

6.5.3.20 Type: VnfLocationConstraint

This type represents the association of location constraints to a VNF instance to be created according to a specific VNF profile. It shall comply with the provisions defined in Table 6.5.3.20-1.

Table 6.5.3.20-1: Definition of the VnfLocationConstraint data type

Attribute name	Data type	Cardinality	Description
vnfProfileId	IdentifierInNsd	1	Identifier (reference to) of a VnfProfile in the NSD used to manage the lifecycle of the VNF instance.
locationConstraints	LocationConstraints	1	Defines the location constraints for the VNF instance to be created based on the VNF profile. See note.
NOTE: These constraints are typically determined by the OSS from service requirements (e.g. latency requirements, regulatory requirements). The NFVO can map such location constraints to eligible NFVI-PoPs/ resource zones where the VNF instance is to be created.			

6.5.3.21 Type: LocationConstraints

This type represents location constraints for a VNF to be instantiated. The location constraints shall be presented as a country code, optionally followed by a civic address based on the format defined by IETF RFC 4776 [13]. The LocationConstraints data type shall comply with the provisions defined in Table 6.5.3.21-1.

Table 6.5.3.21-1: Definition of the LocationConstraints data type

Attribute name	Data type	Cardinality	Description
countryCode	String	1	The two-letter ISO 3166 [29] country code in capital letters.
civicAddressElement	Structure (inlined)	0..N	Zero or more elements comprising the civic address.
>caType	Integer	1	Describe the content type of caValue. The value of caType shall comply with Section 3.4 of IETF RFC 4776 [13].
>caValue	String	1	Content of civic address element corresponding to the caType. The format caValue shall comply with section 3.4 of IETF RFC 4776 [13].

6.5.3.21a Type: ParamsForNestedNs

This type specifies additional parameters on a per-nested NS instance basis. It shall comply with the provisions defined in Table 6.5.3.21a-1.

Table 6.5.3.21a-1: Definition of the ParamsForNestedNs data type

Attribute name	Data type	Cardinality	Description
nsProfileId	IdentifierInNsd	1	Identifier of a NsProfile to which the additional parameters apply.
additionalParam	KeyValuePairs	0..N	Additional parameters that are to be applied on a per nested NS instance.

6.5.3.22 Type: ParamsForVnf

This type defines the additional parameters for the VNF instance to be created associated with an NS instance. It shall comply with the provisions defined in Table 6.5.3.22-1.

Table 6.5.3.22-1: Definition of the ParamsForVnf data type

Attribute name	Data type	Cardinality	Description
vnfProfileId	IdentifierInNsd	1	Identifier of (reference to) a vnfProfile to which the additional parameters apply.
additionalParams	KeyValuePairs	0..1	Additional parameters that are applied for the VNF instance to be created.

6.5.3.23 Type: AffinityOrAntiAffinityRule

This type describes the additional affinity or anti-affinity rule applicable between the VNF instances to be instantiated in the NS instantiation operation request or between the VNF instances to be instantiated in the NS instantiation operation request and the existing VNF instances. It shall comply with the provisions defined in Table 6.5.3.23-1.

Table 6.5.3.23-1: Definition of the AffinityOrAntiAffinityRule data type

Attribute name	Data type	Cardinality	Description
vnfdId	Identifier	0..N	Reference to a VNFD. When the VNFD which is not used to instantiate VNF, it presents all VNF instances of this type as the subjects of the affinity or anti-affinity rule. The VNF instance which the VNFD presents is not necessary as a part of the NS to be instantiated.
vnfProfileId	IdentifierInNsd	1..N	Reference to a vnfProfile defined in the NSD. At least one VnfProfile which is used to instantiate VNF for the NS to be instantiated as the subject of the affinity or anti-affinity rule shall be present. When the VnfProfile which is not used to instantiate VNF, it presents all VNF instances of this type as the subjects of the affinity or anti-affinity rule. The VNF instance which the VnfProfile presents is not necessary as a part of the NS to be instantiated.
vnfInstanceId	Identifier	0..N	Reference to the existing VNF instance as the subject of the affinity or anti-affinity rule. The existing VNF instance is not necessary as a part of the NS to be instantiated.
affinityOrAntiAffinity	Enum (inlined)	1	The type of the constraint. Permitted values: AFFINITY ANTI_AFFINITY
scope	Enum (inlined)	1	Specifies the scope of the rule where the placement constraint applies. Permitted values: NFVI_POP ZONE ZONE_GROUP NFVI_NODE

6.5.3.24 Type: InstantiateVnfData

This type represents the information related to a SAP of a NS. The InstantiateVnfData data type specifies the parameters that are needed for VNF instantiation. This information element is used for the bottom-up NS creation when the OSS/BSS explicitly requests VNF instantiation for a given NS. When the NFVO invokes the Instantiate VNF update operation, a set of these parameters are then passed by the NFVO to the VNFM. It shall comply with the provisions defined in Table 6.5.3.24-1.

Table 6.5.3.24-1: Definition of the InstantiateVnfData data type

Attribute name	Data type	Cardinality	Description
vnfdId	Identifier	1	Information sufficient to identify the VNFD which defines the VNF to be instantiated.
vnfFlavourId	IdentifierInVnfd	1	Identifier of the VNF deployment flavour to be instantiated.
vnfInstantiationLevelId	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.
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.
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.
localizationLanguage	String	0..1	Localization language of the VNF to be instantiated. The value shall comply with the format defined in IETF RFC 5646 [16].
additionalParams	KeyValuePairs	0..1	Additional input parameters for the instantiation process, specific to the VNF being instantiated.

6.5.3.25 Type: ChangeVnfFlavourData

The type represents the information that is requested to be changed deployment flavour for an existing VNF instance. It shall comply with the provisions defined in Table 6.5.3.25-1.

Table 6.5.3.25-1: Definition of the ChangeVnfFlavourData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance to be modified.
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 NFVO.
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".

6.5.3.26 Type: ExtVirtualLinkData

This type represents an external VL. It shall comply with the provisions defined in Table 6.5.3.26-1.

Table 6.5.3.26-1: Definition of the ExtVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
extVirtualLinkId	Identifier	0..1	The identifier of the external VL instance, if provided.
vimId	Identifier	0..1	Identifier of the VIM that manages this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
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.

6.5.3.27 Type: ExtManagedVirtualLinkData

This type represents an externally-managed internal VL. It shall comply with the provisions defined in Table 6.5.3.27-1.

Table 6.5.3.27-1: Definition of the ExtManagedVirtualLinkData data type

Attribute name	Data type	Cardinality	Description
extManagedVirtualLinkId	Identifier	0..1	The identifier of the externally-managed internal VL instance, if provided.
vmfVirtualLinkDescId	IdentifierInVnfd	1	The identifier of the VLD in the VNFD for this VL.
vimId	Identifier	0..1	Identifier of the VIM that manage this resource. This attribute shall only be supported and present if VNF-related resource management in direct mode is applicable.
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.

6.5.3.28 Type: ExtLinkPortData

This type represents an externally provided link port to be used to connect a VNF external connection point to an external VL. It shall comply with the provisions defined in Table 6.5.3.28-1.

Table 6.5.3.28-1: 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.

6.5.3.29 Type: VnfExtCpData

This type represents configuration information for external CPs created from a CPD. It shall comply with the provisions defined in Table 6.5.3.29-1.

Table 6.5.3.29-1: 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.

6.5.3.30 Type: VnfExtCpConfig

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

This type shall comply with the provisions defined in Table 6.5.3.30-1.

Table 6.5.3.30-1: 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": – The "linkPortId" and "cpProtocolData" attributes shall both be absent for the deletion of an existing external CP instance addressed by cpInstanceId. – At least one of these attributes shall be present for a to-be-created external CP instance or an existing external CP instance.			

6.5.3.31 Type: OperateVnfData

This type represents a VNF instance for which the operational state needs to be changed and the requested new state. It shall comply with the provisions defined in Table 6.5.3.31-1.

Table 6.5.3.31-1: Definition of the OperateVnfData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance.
changeStateTo	OperationalStates	1	The desired operational state (i.e. started or stopped) to change the VNF to.
stopType	StopType	0..1	It signals whether forceful or graceful stop is requested. See note.
gracefulStopTimeout	Integer	0..1	The time interval (in seconds) to wait for the VNF to be taken out of service during graceful stop, before stopping the VNF. See note.
NOTE: The "stopType" and "gracefulStopTimeout" attributes shall be absent, when the "changeStateTo" attribute is equal to "STARTED". The "gracefulStopTimeout" attribute shall be present, when the "changeStateTo" attribute is equal to "STOPPED" and the "stopType" attribute is equal to "GRACEFUL". The "gracefulStopTimeout" attribute shall be absent, when the "changeStateTo" attribute is equal to "STOPPED" and the "stopType" attribute is equal to "FORCEFUL". The request shall be treated as if the "stopType" attribute was set to "FORCEFUL", when the "changeStateTo" attribute is equal to "STOPPED" and the "stopType" attribute is absent.			

6.5.3.32 Type: ModifyVnfInfoData

This type represents the information that is requested to be modified for a VNF instance. The information to be modified shall comply with the associated NSD.

EXAMPLE: The vnfPkgId attribute value for a particular VNF instance can only be updated with a value that matches the identifier value of a VNF package whose vnfdId is present in the associated profile of the NSD.

Thus type shall comply with the provisions defined in Table 6.5.3.32-1.

Table 6.5.3.32-1: Definition of the ModifyVnfInfoData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance.
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 to entries in the "vnfConfigurableProperties" list, as defined below this Table.
Metadata	KeyValuePairs	0..1	Modifications to entries in the "metadata" list, as defined below this Table.
Extensions	KeyValuePairs	0..1	Modifications to entries in the "extensions" list, as defined below this Table.

6.5.3.33 Type: ChangeExtVnfConnectivityData

This type describes the information invoked by the NFVO to change the external VNF connectivity information maintained by the VNFM. The types of changes that this operation supports are:

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

2) 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.

This type shall comply with the provisions defined in Table 6.5.3.33-1.

Table 6.5.3.33-1: Definition of the ChangeExtVnfConnectivityData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifier of the VNF instance.
extVirtualLink	ExtVirtualLinkData	1..N	Information about external VLs to change (e.g. connect the VNF to).
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the OSS as input to the external connectivity change process, specific to the VNF instance being changed.

6.5.3.34 Type: AssocNewNsdVersionData

This type specifies a new NSD version that is associated to the NS instance. After issuing the Update NS operation with updateType = "AssocNewNsdVersion", the NFVO shall use the referred NSD as a basis for the given NS instance. Different versions of the same NSD have same nsdInvariantId, but different nsdId attributes, therefore if the nsdInvariantId of the NSD version that is to be associated to this NS instance is different from the one used before, the NFVO shall reject the request. Only new versions of the same NSD can be associated to an existing NS instance. This data type shall comply with the provisions defined in Table 6.5.3.34-1.

Table 6.5.3.34-1: Definition of the AssocNewNsdVersionData data type

Attribute name	Data type	Cardinality	Description
newNsdId	Identifier	1	Identifier of the new NSD version that is to be associated to the NS instance.
sync	Boolean	0..1	Specify whether the NS instance shall be automatically synchronized to the new NSD by the NFVO (in case of true value) or the NFVO shall not do any action (in case of a false value) and wait for further guidance from OSS/BSS (i.e. waiting for OSS/BSS to issue NS lifecycle management operation to explicitly add/remove VNFs and modify information of VNF instances according to the new NSD). The synchronization to the new NSD means e.g. instantiating/adding those VNFs whose VNFD is referenced by the new NSD version but not referenced by the old one, terminating/removing those VNFs whose VNFD is referenced by the old NSD version but not referenced by the new NSD version, modifying information of VNF instances to the new applicable VNFD provided in the new NSD version. A cardinality of 0 indicates that synchronization shall not be done.

6.5.3.35 Type: MoveVnfInstanceData

This type specifies existing VNF instances to be moved from one NS instance (source) to another NS instance (destination). The NS instance defined in the Update NS operation indicates the source NS instance and the destination NS instance is specified in this data type (referred to targetNsInstanceId). It shall comply with the provisions defined in Table 6.5.3.35-1.

Table 6.5.3.35-1: Definition of the MoveVnfInstanceData data type

Attribute name	Data type	Cardinality	Description
targetNsInstanceId	Identifier	1	Specify the target NS instance where the VNF instances are moved to.
vnfInstanceId	Identifier	1..N	Specify the VNF instance that is moved.

6.5.3.36 Type: AddVnffgData

This type specifies the parameters used for the creation of a new VNFFG instance. It shall comply with the provisions defined in Table 6.5.3.36-1.

Table 6.5.3.36-1: Definition of the AddVnffgData data type

Attribute name	Data type	Cardinality	Description
vnffgdId	IdentifierInNsd	1	Identifier of the VNFFGD used to create this VNFFG instance.
vnffgName	String	1	Human readable name for the VNFFG.
description	String	1	Human readable description for the VNFFG.

6.5.3.37 Type: UpdateVnffgData

This type specifies the parameters used for the update of an existing VNFFG instance. It shall comply with the provisions defined in Table 6.5.3.37-1.

Table 6.5.3.37-1: Definition of the UpdateVnffgData data type

Attribute name	Data type	Cardinality	Description
vnffgInfoId	IdentifierInNs	1	Identifier of an existing VNFFG to be updated for the NS Instance.
nfp	NfpData	0..N	Indicate the desired new NFP(s) for a given VNFFG after the operations of addition/removal of NS components (e.g. VNFs, VLs, etc.) have been completed, or indicate the updated or newly created NFP classification and selection rule which applied to an existing NFP.
nfpInfoId	IdentifierInNs	0..N	Identifier(s) of the NFP to be deleted from a given VNFFG.

6.5.3.38 Type: NfpData

This type contains information used to create or modify NFP instance parameters for the update of an existing VNFFG instance. It shall comply with the provisions defined in Table 6.5.3.38-1.

Table 6.5.3.38-1: Definition of the NfpData data type

Attribute name	Data type	Cardinality	Description
nfpInfoId	IdentifierInNs	0..1	Identifier of the NFP to be modified. It shall be present for modified NFPs and shall be absent for the new NFP. See note 1.
nfpName	String	0..1	Human readable name for the NFP. It shall be present for the new NFP, and it may be present otherwise. See note 2.
description	String	0..1	Human readable description for the NFP. It shall be present for the new NFP, and it may be present otherwise. See note 2.
cpGroup	CpGroupInfo	0..N	Group(s) of CPs and/or SAPs which the NFP passes by. Cardinality can be 0 if only updated or newly created NFP classification and selection rule which applied to an existing NFP is provided. See notes 3 and 4.
nfpRule	NfpRule	0..1	NFP classification and selection rule. See note 1.
NOTE 1: It shall be present for modified NFPs and shall be absent for the new NFP. NOTE 2: It shall be present for the new NFP, and it may be present otherwise. NOTE 3: At least a CP or an nfpRule shall be present. NOTE 4: When multiple identifiers are included, the position of the identifier in the cpGroup value specifies the position of the group in the path.			

6.5.3.39 Type: ChangeNsFlavourData

This type specifies an existing NS instance for which the DF needs to be changed. This specifies the new DF, the instantiationLevel of the new DF that may be used and the additional parameters as input for the flavour change. It shall comply with the provisions defined in Table 6.5.3.39-1.

Table 6.5.3.39-1: Definition of the ChangeNsFlavourData data type

Attribute name	Data type	Cardinality	Description
newNsFlavourId	IdentifierInNsd	1	Identifier of the new NS DF to apply to this NS instance.
instantiationLevelId	IdentifierInNsd	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 NSD is instantiated.

6.5.3.40 Type: NfpRule

The NfpRule data type is an expression of the conditions that shall be met in order for the NFP to be applicable to the packet. The condition acts as a flow classifier and it is met only if all the values expressed in the condition are matched by those in the packet. It shall comply with the provisions defined in Table 6.5.3.40-1.

Table 6.5.3.40-1: Definition of the NfpRule data type

Attribute name	Data type	Cardinality	Description
etherDestinationAddress	MacAddress	0..1	Indicates a destination Mac address See note.
etherSourceAddress	MacAddress	0..1	Indicates a source Mac address See note.
etherType	Enum (inlined)	0..1	Indicates the protocol carried over the Ethernet layer. Permitted values: IPV4 IPV6 See note.
vlanTag	String	0..N	Indicates a VLAN identifier in an IEEE 802.1Q-2014 tag [6] Multiple tags can be included for QinQ stacking. See note.
protocol	Enum (inlined)	0..1	Indicates the L4 protocol, For IPv4 [7] this corresponds to the field called "Protocol" to identify the next level protocol. For IPv6 [28] this corresponds to the field is called the "Next Header" field. Permitted values: Any keyword defined in the IANA protocol registry [1], e.g.: - TCP - UDP - ICMP See note.
dscp	String	0..1	For IPv4 [7] a string of "0" and "1" digits that corresponds to the 6-bit Differentiated Services Code Point (DSCP) field of the IP header. For IPv6 [28] a string of "0" and "1" digits that corresponds to the 6 differentiated services bits of the traffic class header field. See note.
sourcePortRange	PortRange	0..1	Indicates a range of source ports. See note.
destinationPortRange	PortRange	0..1	Indicates a range of destination ports. See note.
sourceIpAddressPrefix	IpAddressPrefix	0..1	Indicates the source IP address range in CIDR format. See note.
destinationIpAddressPrefix	IpAddressPrefix	0..1	Indicates the destination IP address range in CIDR format. See note.
extendedCriteria	Mask	0..N	Indicates values of specific bits in a frame. See note.
NOTE: At least one attribute shall be present. If multiple attributes are present, a logical "AND" operation shall be applied to those attributes when matching packets against the rule.			

6.5.3.41 Type: Mask

The Mask data type identifies the value to be matched for a sequence of bits at a particular location in a frame. It shall comply with the provisions defined in Table 6.5.3.41-1.

Table 6.5.3.41-1: Definition of the Mask data type

Attribute name	Data type	Cardinality	Description
startingPoint	Integer	1	Indicates the offset between the last bit of the source mac address and the first bit of the sequence of bits to be matched.
length	Integer	1	Indicates the number of bits to be matched.
value	String	1	Provide the sequence of bit values to be matched.

6.5.3.42 Type: PortRange

The PortRange data type provides the lower and upper bounds of a range of Internet ports. It shall comply with the provisions defined in Table 6.5.3.42-1.

Table 6.5.3.42-1: Definition of the PortRange data type

Attribute name	Data type	Cardinality	Description
lowerPort	Integer	1	Identifies the lower bound of the port range.
upperPort	Integer	1	Identifies the upper bound of the port range

6.5.3.43 Type: HealNsData

This type represents the information used to heal a NS. It shall comply with the provisions defined in Table 6.5.3.43-1.

Table 6.5.3.43-1: Definition of the HealNsData data type

Attribute name	Data type	Cardinality	Description
degreeHealing	Enum (inlined)	1	Indicates the degree of healing. Possible values include: - HEAL_RESTORE: Complete the healing of the NS restoring the state of the NS before the failure occurred - HEAL_QOS: Complete the healing of the NS based on the newest QoS values - HEAL_RESET: Complete the healing of the NS resetting to the original instantiation state of the NS - PARTIAL_HEALING
actionsHealing	String	0..N	Used to specify dedicated healing actions in a particular order (e.g. as a script). The actionsHealing attribute can be used to provide a specific script whose content and actions might only be possible to be derived during runtime. See note.
healScript	IdentifierInNsd	0..1	Reference to a script from the NSD that shall be used to execute dedicated healing actions in a particular order. The healScript, since it refers to a script in the NSD, can be used to execute healing actions which are defined during NS design time. See note.
additionalParamsforNs	KeyValuePairs	0..1	Allows the OSS/BSS to provide additional parameter(s) to the healing process at the NS level.
NOTE: Either the actionsHealing or healScript attribute shall be present, not both attributes.			

6.5.3.44 Type: HealVnfData

This type represents the information to heal a VNF that is part of an NS. The NFVO shall then invoke the HealVNF operation towards the appropriate VNFM. It shall comply with the provisions defined in Table 6.5.3.44-1.

Table 6.5.3.44-1: Definition of the HealVnfData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	1	Identifies the VNF instance, part of the NS, 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 NFVO as input to the healing process, specific to the VNF being healed. EXAMPLE: Input parameters to VNF-specific healing procedures.

6.5.3.45 Type: ScaleNsData

This type represents the information to scale a NS. It shall comply with the provisions defined in Table 6.5.3.45-1.

Table 6.5.3.45-1: Definition of the ScaleNsData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceToBeAdded	VnfInstanceData	0..N	An existing VNF instance to be added to the NS instance as part of the scaling operation. If needed, the VNF Profile to be used for this VNF instance may also be provided. See notes 1, 2 and 3.
vnfInstanceToBeRemoved	Identifier	0..N	The VNF instance to be removed from the NS instance as part of the scaling operation. See notes 1 and 4.
scaleNsByStepsData	ScaleNsByStepsData	0..1	The information used to scale an NS instance by one or more scaling steps. See note 1.
scaleNsToLevelData	ScaleNsToLevelData	0..1	The information used to scale an NS instance to a target size. See note 1.
additionalParamsForNs	KeyValuePairs	0..1	Allows the OSS/BSS to provide additional parameter(s) at the NS level necessary for the NS scaling (as opposed to the VNF level, which is covered in additionalParamForVnf).
additionalParamsForVnf	ParamsForVnf	0..N	Allows the OSS/BSS to provide additional parameter(s) per VNF instance (as opposed to the NS level, which is covered in additionalParamforNs). This is for VNFs that are to be created by the NFVO as part of the NS scaling and not for existing VNF that are covered by the scaleVnfData.
locationConstraints	VnfLocationConstraint	0..N	The location constraints for the VNF to be instantiated as part of the NS scaling. An example can be a constraint for the VNF to be in a specific geographic location.
NOTE 1: No more than two attributes between vnfInstanceToBeAdded, vnfInstanceToBeRemoved, scaleNsByStepsData and scaleNsToLevelData shall be present. In case of two, the attributes shall be vnfInstanceToBeAdded and vnfInstanceToBeRemoved. NOTE 2: The DF of the VNF instance shall match the VNF DF present in the associated VNF Profile of the new NS flavour. NOTE 3: This functionality is the same as the one provided by the Update NS operation when the AddVnf update type is selected (see clause 7.3.5). NOTE 4: This functionality is the same as the one provided by the Update NS operation when the RemoveVnf update type is selected (see clause 7.3.5).			

6.5.3.46 Type: ScaleNsByStepsData

This type represents the information used to scale an NS instance by one or more scaling steps, with respect to a particular NS scaling aspect. Performing a scaling step means increasing/decreasing the capacity of an NS instance in a discrete manner, i.e. moving from one NS scale level to another. The NS scaling aspects and their corresponding NS scale levels applicable to the NS instance are declared in the NSD. It shall comply with the provisions defined in Table 6.5.3.46-1.

Table 6.5.3.46-1: Definition of the ScaleNsByStepsData data type

Attribute name	Data type	Cardinality	Description
scalingDirection	Enum (inlined)	1	The scaling direction. Possible values are: - SCALE_IN - SCALE_OUT.
aspectId	IdentifierInNsd	1	The aspect of the NS that is requested to be scaled, as declared in the NSD.
numberOfSteps	Integer	0..1	The number of scaling steps to be performed. Defaults to 1.

6.5.3.47 Type: ScaleNsToLevelData

This type represents the information used to scale an NS instance to a target size. The target size is either expressed as an NS instantiation level or as a list of NS scale levels, one per NS scaling aspect, of the current DF. The NS instantiation levels, the NS scaling aspects and their corresponding NS scale levels applicable to the NS instance are declared in the NSD. It shall comply with the provisions defined in Table 6.5.3.47-1.

Table 6.5.3.47-1: Definition of the ScaleNsToLevelData data type

Attribute name	Data type	Cardinality	Description
nsInstantiationLevel	IdentifierInNsd	0..1	Identifier of the target NS instantiation level of the current DF to which the NS instance is requested to be scaled. See note.
nsScaleInfo	NsScaleInfo	0..N	For each NS scaling aspect of the current DF, defines the target NS scale level to which the NS instance is to be scaled. See note.
NOTE: Either nsInstantiationLevel or nsScaleInfo, but not both, shall be present.			

6.5.3.48 Type: NsScaleInfo

This type represents the target NS Scale level for each NS scaling aspect of the current deployment flavour. It shall comply with the provisions defined in Table 6.5.3.48-1.

Table 6.5.3.48-1: Definition of the NsScaleInfo data type

Attribute name	Data type	Cardinality	Description
nsScalingAspectId	IdentifierInNsd	1	Identifier of the NS scaling aspect.
nsScaleLevelId	IdentifierInNsd	1	Identifier of the NS scale level.

6.5.3.49 Type: ScaleVnfData

This type represents defines the information to scale a VNF instance to a given level, or to scale a VNF instance by steps. It shall comply with the provisions defined in Table 6.5.3.49-1.

Table 6.5.3.49-1: Definition of the ScaleVnfData data type

Attribute name	Data type	Cardinality	Description
vnfInstanceid	Identifier	1	Identifier of the VNF instance being scaled.
scaleVnfType	Enum (inlined)	1	Type of the scale VNF operation requested. Allowed values are: - SCALE_OUT - SCALE_IN - SCALE_TO_INSTANTIATION_LEVEL - SCALE_TO_SCALE_LEVEL(S) The set of types actually supported depends on the capabilities of the VNF being managed. See note 1.
scaleToLevelData	ScaleToLevelData	0..1	The information used for scaling to a given level. See note 2.
scaleByStepData	ScaleByStepData	0..1	The information used for scaling by steps. See note 2.
NOTE 1: ETSI GS NFV-IFA 010 [2] specifies that the lifecycle management operations that expand or contract a VNF instance include scale in, scale out, scale up and scale down. Vertical scaling (scale up, scale down) is not supported in the present document. NOTE 2: Either scaletoLevelData or scaleByStepData but not both shall be present. The scaleByStepData is used for scale out/in type of scaling, and the scaleToLevelData is used for scale to instantiation/scale level type of scaling.			

6.5.3.50 Type: ScaleToLevelData

This type describes the information used to scale a VNF instance to a target size. The target size is either expressed as an instantiation level of that DF as defined in the VNFD, or given as a list of scale levels, one per scaling aspect of that DF. Instantiation levels and scaling aspects are declared in the VNFD. The NFVO shall then invoke the ScaleVnfToLevel operation towards the appropriate VNFM. It shall comply with the provisions defined in Table 6.5.3.50-1.

Table 6.5.3.50-1: Definition of the ScaleToLevelData data type

Attribute name	Data type	Cardinality	Description
vnfInstantiationLevelId	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.
vnfScaleInfo	VnfScaleInfo	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 NFVO as input to the scaling process, specific to the VNF being scaled.
NOTE: Either the instantiationLevelId attribute or the scaleInfo attribute shall be included.			

6.5.3.51 Type: VnfScaleInfo

This type describes the provides information about the scale level of a VNF instance with respect to one scaling aspect. It shall comply with the provisions defined in Table 6.5.3.51-1.

Table 6.5.3.51-1: Definition of the VnfScaleInfo data type

Attribute name	Data type	Cardinality	Description
aspectlId	IdentifierInVnfd	1	The scaling aspect.
scaleLevel	Integer	1	The scale level for that aspect. Minimum value 0, maximum value maxScaleLevel as declared in the VNFD.

6.5.3.52 Type: ScaleByStepData

This type describes the information to scale a VNF instance by steps. The NFVO shall then invoke the Scale VNF operation towards the appropriate VNFM. It shall comply with the provisions defined in Table 6.5.3.52-1.

Table 6.5.3.52-1: Definition of the ScaleByStepData data type

Attribute name	Data type	Cardinality	Description
aspectId	IdentifierInVnfd	1	Identifier of (reference to) the aspect of the VNF that is requested to be scaled, as declared in the VNFD.
numberOfSteps	Integer	0..1	Number of scaling steps. It shall be a positive number. Defaults to 1. The VNF provider defines in the VNFD whether or not a particular VNF supports performing more than one step at a time. Such a property in the VNFD applies for all instances of a particular VNF. See note.
additionalParams	KeyValuePairs	0..1	Additional parameters passed by the NFVO as input to the scaling process, specific to the VNF instance being scaled.
NOTE: A scaling step is the smallest unit by which a VNF instance can be scaled w.r.t a particular scaling aspect.			

6.5.3.53 Type: NsVirtualLinkInfo

This type specifies the information about an NS VL instance. It shall comply with the provisions defined in Table 6.5.3.53-1.

Table 6.5.3.53-1: Definition of the NsVirtualLinkInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInNs	1	Identifier of the VL instance.
nsVirtualLinkDescId	IdentifierInNsd	1	Identifier of the VLD in the NSD.
nsVirtualLinkProfileId	IdentifierInNsd	1	Identifier of the VL profile in the NSD.
resourceHandle	ResourceHandle	1..N	Identifier(s) of the virtualised network resource(s) realizing the VL instance. See note.
linkPort	NsLinkPortInfo	0..N	Link ports of the VL instance. Cardinality of zero indicates that no port has yet been created for the VL instance.
NOTE: As an NS can include NFs deployed in NFVI PoPs under the control of several different VIMs, deploying an NS VL can involve several VIMs each allocating different virtualised network resources. When this NsVirtualLink is provided as an ExtVirtualLink as input of a VNF LCM operation, the id of the ExtVirtualLink shall be the same as the corresponding NsVirtualLink. The connectivity between virtualised network resources allocated in different VIMs and part of the same VL is not addressed in the present document.			

6.5.3.54 Void

6.5.3.55 Type: NsLinkPortInfo

This type represents information about a link port of a VL instance. It shall comply with the provisions defined in Table 6.5.3.55-1.

Table 6.5.3.55-1: Definition of the NsLinkPortInfo 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	Identifier of the virtualised network resource realizing this link port.
nsCpHandle	NsCpHandle	0..1	Identifier of the CP/SAP instance to be connected to this link port. The value refers to a vnfExtCpInfo item in the VnfInstance, or a pnfExtCpInfo item in the PnfInfo, or a sapInfo item in the NS instance. There shall be at most one link port associated with any connection point instance.
NOTE: When the NsVirtualLink, from which the present NsLinkPort is part of, is provided as an ExtVirtualLink as input of a VNF LCM operation, the id of the ExtLinkPort shall be the same as the corresponding NsLinkPort.			

6.5.3.56 Type: NsCpHandle

This type represents an identifier of the CP or SAP instance. It shall comply with the provisions defined in Table 6.5.3.56-1.

Table 6.5.3.56-1: Definition of the NsCpHandle data type

Attribute name	Data type	Cardinality	Description
vnfInstanceId	Identifier	0..1	Identifier of the VNF instance associated to the CP instance. This attribute shall be present if the CP instance is VNF external CP. See notes 1 and 4.
vnfExtCpInstanceId	IdentifierInVnf	0..1	Identifier of the VNF external CP instance in the scope of the VNF instance. This attribute shall be present if the CP instance is VNF external CP. See notes 1 and 4.
pnfInfoId	Identifier	0..1	Identifier of the PNF instance associated to the CP instance. This attribute shall be present if the CP instance is PNF external CP. See notes 2 and 4.
pnfExtCpInstanceId	IdentifierInPnf	0..1	Identifier of the PNF external CP instance in the scope of the PNF. This attribute shall be present if the CP instance is PNF external CP. See notes 2 and 4.
nsInstanceId	Identifier	0..1	Identifier of the NS instance associated to the SAP instance. This attribute shall be present if the CP instance is NS SAP. See notes 3 and 4.
nsSapInstanceId	IdentifierInNs	0..1	Identifier of the SAP instance in the scope of the NS instance. This attribute shall be present if the CP instance is NS SAP. See notes 3 and 4.
NOTE 1: For the VNF external CP instance, both vnfInstanceId and vnfExtCpInstanceId shall be present as a pair. NOTE 2: For the PNF external CP instance, both pnfInfoId and PnfExtCpInstanceId shall be present as a pair. NOTE 3: For the SAP instance, both nsInstanceId and nsSapInstanceId shall be present as a pair. NOTE 4: One pair of identifiers (VNF external CP, PNF external CP or SAP) shall be present.			

6.5.3.57 Type: VnfInstance

This type represents a VNF instance. It shall comply with the provisions defined in Table 6.5.3.57-1.

NOTE: Clause B.3.2 of ETSI GS NFV-SOL 003 [4] 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 6.5.3.57-1: 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.

Attribute name	Data type	Cardinality	Description
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 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. - 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.
vimId	Identifier	0..N	Identifier of a VIM that manages resources for the VNF instance.
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	VnfScaleInfo	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 of ETSI GS NFV-SOL 003 [4] for an explanation of VNF scaling.
>extCpInfo	VnfExtCpInfo	1..N	Information about the external CPs exposed by the VNF instance.
>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	VnfMonitoringParameter	0..N	Performance metrics tracked by the VNFM (e.g. for auto-scaling purposes) as identified by the VNF provider in the VNFD.

Attribute name	Data type	Cardinality	Description
>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 [16].
>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.
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.
NOTE 1: Modifying the value of this attribute shall not be performed when no 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.3] 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.			

6.5.3.58 Type: CpProtocolInfo

This type describes the protocol layer(s) that a CP or SAP uses together with protocol-related information, like addresses. It shall comply with the provisions defined in Table 6.5.3.58-1.

Table 6.5.3.58-1: 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 CP or SAP 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.			

6.5.3.59 Type: ExtManagedVirtualLinkInfo

This type provides information about an externally-managed virtual link for VNFs. It shall comply with the provisions defined in Table 6.5.3.59-1.

Table 6.5.3.59-1: 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.
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.

6.5.3.60 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 6.5.3.60-1.

Table 6.5.3.60-1: 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. Shall be present when that particular CP of the VNFC instance is associated to an external CP of the VNF instance. May be present otherwise.
>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 CP.
metadata	KeyValuePairs	0..1	Metadata about this resource.
NOTE: ETSI GS NFV-SOL 001 [i.3] specifies the structure and format of the VNFD based on TOSCA specifications.			

6.5.3.61 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 6.5.3.61-1.

Table 6.5.3.61-1: 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.

6.5.3.62 Type: ExtVirtualLinkInfo

This type represents information about an VNF external VLs. It shall comply with the provisions defined in Table 6.5.3.62-1.

Table 6.5.3.62-1: 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.
resourceHandle	ResourceHandle	1	Reference to the resource realizing this VL.
extLinkPorts	ExtLinkPortInfo	0..N	Link ports of this VL.

6.5.3.63 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 6.5.3.63-1.

Table 6.5.3.63-1: 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 VNFconnected 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.

6.5.3.64 Type: VnfLinkPortInfo

This type represents a link port of an internal VL of a VNF. It shall comply with the provisions defined in Table 6.5.3.64-1.

Table 6.5.3.64-1: 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 associated with this link port. When the link port is used for internal connectivity in the VNF, this attribute represents the VNFC CP to be connected to this link port. Shall be present when the link port is used for external connectivity by the VNF. May be present if used to reference a VNFC CP instance. 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 "vnfcResouceInfo" item in the VnfInstance.
cpInstanceType	Enum (inlined)	0..1	Type of the CP instance that is identified by cpInstanceId. Shall be present if "cpInstanceId" is present, and shall be absent otherwise. Permitted values: VNFC_CP: The link port is connected to a VNFC CP EXT_CP: The link port is associated to an external CP.

6.5.3.65 Type: VnffgInfo

This type specifies the information about a VNFFG instance. It shall comply with the provisions defined in Table 6.5.3.65-1.

Table 6.5.3.65-1: Definition of the VnffgInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this VNFFG instance.
vnffgdId	IdentifierInNsd	1	Identifier of the VNFFGD in the NSD.
vnfInstanceId	Identifier	1..N	Identifier(s) of the constituent VNF instance(s) of this VNFFG instance.
pnfInfoId	Identifier	0..N	Identifier(s) of the constituent PNF instance(s) of this VNFFG instance.
nsVirtualLinkInfoId	IdentifierInNs	1..N	Identifier(s) of the constituent VL instance(s) of this VNFFG instance.
nsCpHandle	NsCpHandle	1..N	Identifiers of the CP instances attached to the constituent VNFs and PNFs or the SAP instances of the VNFFG. See note.
nfpInfo	NfpInfo	1..N	Information on the NFP instances.
NOTE: It indicates an exhaustive list of all the CP instances and SAP instances of the VNFFG.			

6.5.3.66 Type: NfpInfo

This type represents an NFP instance. It shall comply with the provisions defined in Table 6.5.3.66-1.

Table 6.5.3.66-1: Definition of the NfpInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInNs	1	Identifier of this NFP instance.
nfpdId	IdentifierInNsd	0..1	Identifier of the NFPD used to instantiate this NFP instance. It shall be present if the NFP instance is instantiated from the NFPD.
nfpName	String	0..1	Human readable name for the NFP instance.
description	String	0..1	Human readable description for the NFP instance.
cpGroup	CpGroupInfo	1..N	Group(s) of CPs and/or SAPs which the NFP passes through. See note.
totalCp	Integer	0..1	Total number of CP and SAP instances in this NFP instance.
nfpRule	NfpRule	1	NFP classification and selection rule.
nfpState	Enum (inlined)	1	The state of the NFP instance. Permitted values: ENABLED: The NFP instance is enabled. DISABLED: The NFP instance is disabled.
NOTE: When multiple identifiers are included, the position of the identifier in the CpGroup data type specifies the position of the group in the path.			

6.5.3.67 Type: SapInfo

This type represents an SAP instance. It shall comply with the provisions defined in Table 6.5.3.67-1.

Table 6.5.3.67-1: Definition of the SapInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInNs	1	Identifier of the SAP instance.
sapdId	IdentifierInNsd	1	Identifier of the SAPD in the NSD.
sapName	String	1	Human readable name for the SAP instance.
description	String	1	Human readable description for the SAP instance.
sapProtocolInfo	CpProtocolInfo	1..N	Network protocol information for this SAP.

6.5.3.68 Type: NsMonitoringParameter

This type represents a monitoring parameter that is tracked by the NFVO, for example, for auto-scaling purposes. It shall comply with the provisions defined in Table 6.5.3.68-1.

Table 6.5.3.68-1: Definition of the NsMonitoringParameter data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInNsd	1	Identifier of the monitoring parameter defined in the NSD.
name	String	0..1	Human readable name of the monitoring parameter, as defined in the NSD.
performanceMetric	String	1	Performance metric that is monitored. This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [31].

6.5.3.69 Type: VnfMonitoringParameter

This type represents a monitoring parameter that is tracked by the VNFM, for example, for auto-scaling purposes. It shall comply with the provisions defined in Table 6.5.3.69-1.

Table 6.5.3.69-1: Definition of the VnfMonitoringParameter 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.
performanceMetric	String	1	Performance metric that is monitored. This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [31].

6.5.3.70 Type: VnfExtCpInfo

This type represents information about an external CP of a VNF. It shall comply with the provisions defined in Table 6.5.3.70-1.

Table 6.5.3.70-1: Definition of the VnfExtCpInfo data type

Attribute name	Data type	Cardinality	Description
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	1..N	Network protocol information for this CP.
extLinkPortId	Identifier	0..1	Identifier of the "extLinkPortInfo" structure inside the "extVirtualLinkInfo" structure. Shall be present if the CP is associated to a link port.
metadata	KeyValuePairs	0..1	Metadata about this external CP.
associatedVnfcCpId	IdentifierInVnf	0..1	Identifier of the "vnfcCpInfo" structure in "VnfcResourceInfo" structure that represents the VNFC CP which is exposed by this external CP instance. Shall be present in case this CP instance maps to a VNFC CP See note.
associatedVnfVirtualLinkId	IdentifierInVnf	0..1	Identifier of the "VnfVirtualLinkResourceInfo" structure that represents the internal VL, which is exposed by this external CP instance. Shall be present in case this CP instance maps to an internal VL. See note.
NOTE: The attributes "associatedVnfcCpId" and "associatedVnfVirtualLinkId" are mutually exclusive. One and only one shall be present.			

6.5.3.71 Type: CpGroupInfo

This type represents describes a group of CPs and/or SAPs pairs associated to the same position in an NFP. It shall comply with the provisions defined in Table 6.5.3.71-1.

Table 6.5.3.71-1: Definition of the CpGroupInfo data type

Attribute name	Data type	Cardinality	Description
cpPairInfo	CpPairInfo	1..N	One or more pair(s) of ingress and egress CPs or SAPs which the NFP passes by. See note.
forwardingBehaviour	Enum (inlined)	0..1	Identifies a rule to apply to forward traffic to the ingress CPs or SAPs of the group. Permitted values: ALL = Traffic flows shall be forwarded simultaneously to all CPs or SAPs of the group. LB = Traffic flows shall be forwarded to one CP or SAP of the group selected based on a load-balancing algorithm.
forwardingBehaviourInputParameters	ForwardingBehaviourInputParameters	0..1	Provides input parameters to configure the forwarding behaviour (e.g. identifies a load balancing algorithm and criteria).
NOTE: All CP or SAP pairs in a group shall be instantiated from connection point descriptors or service access point descriptors referenced in the corresponding NfpPositionDesc.			

6.5.3.72 Type: CpPairInfo

This type represents describes a pair of ingress and egress CPs or SAPs which the NFP passes by. It shall comply with the provisions defined in Table 6.5.3.72-1.

Table 6.5.3.72-1: Definition of the CpPairInfo data type

Attribute name	Data type	Cardinality	Description
vnfExtCpIds	IdentifierInVnf	0..2	Identifier(s) of the VNF CP(s) which form the pair. See notes 1 and 2.
pnfExtCpIds	IdentifierInPnf	0..2	Identifier(s) of the PNF CP(s) which form the pair. See notes 1 and 2.
sapIds	IdentifierInNs	0..2	Identifier(s) of the SAP(s) which form the pair. See notes 1 and 2.
NOTE 1: The presence of a single vnfExpCpId, pnfExtCpId, or sapId occurrence indicates that the CP or SAP is used both as an ingress and egress port at a particular NFP position. NOTE 2: Only one of these three attributes shall be present.			

6.5.3.73 Type: ForwardingBehaviour InputParameters

This type represents provides input parameters to configure the forwarding behaviour. It shall comply with the provisions defined in Table 6.5.3.73-1.

Table 6.5.3.73-1: Definition of the ForwardingBehaviour InputParameters data type

Attribute name	Data type	Cardinality	Description
algortihmName	Enum (Inlined)	0..1	May be included if forwarding behaviour is equal to LB. Shall not be included otherwise. Permitted values: - ROUND_ROBIN - LEAST_CONNECTION - LEAST_TRAFFIC - LEAST_RESPONSE_TIME - CHAINED_FAILOVER - SOURCE_IP_HASH - SOURCE_MAC_HASH
algorithmWeights	Integer	0..N	Percentage of messages sent to a CP instance. May be included if applicable to the algorithm. See notes 1 and 2.
NOTE 1: If applicable to the algorithm but not provided, default values determined by the VIM or NFVI are expected to be used. NOTE 2: Weight applies to the CP instances in the order they have been created.			

6.5.4 Referenced simple data types and enumerations

6.5.4.1 Introduction

This clause defines simple data types 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 4.4.2.

6.5.4.3 Enumeration: NsLcmOpType

The enumeration NsLcmOpType represents those lifecycle operations that trigger a NS lifecycle management operation occurrence notification. It shall comply with the provisions defined in Table 6.5.4.3-1.

Table 6.5.4.3-1: Enumeration NsLcmOpType

Enumeration value	Description
INSTANTIATE	Represents the "Instantiate NS" LCM operation.
SCALE	Represents the "Scale NS" LCM operation.
UPDATE	Represents the "Update NS" LCM operation.
TERMINATE	Represents the "Terminate NS" LCM operation.
HEAL	Represents the "Heal NS" LCM operation.

6.5.4.4 Enumeration: NsLcmOperationStateType

The enumeration NsLcmOperationStateType shall comply with the provisions defined in Table 6.5.4.4-1. More information of the meaning of the states can be found in clause 6.6.2.2.

Table 6.5.4.4-1: Enumeration NsLcmOperationStateType

Enumeration value	Description
PROCESSING	The LCM operation is currently in execution.
COMPLETED	The LCM operation has been completed successfully.
PARTIALLY_COMPLETED	The LCM operation has been partially completed with acceptable errors.
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 NS prior to the original operation invocation has been restored as closely as possible.

6.5.4.5 Enumeration: NsComponentType

The enumeration NsComponentType represents the NS component type. It shall comply with the provisions defined in Table 6.5.4.5-1.

Table 6.5.4.5-1: Enumeration NsComponentType

Enumeration value	Description
VNF	Represents the impacted NS component is a VNF.
PNF	Represents the impacted NS component is a PNF.
NS	Represents the impacted NS component is a nested NS.

6.5.4.6 Enumeration: LcmOpNameForChangeNotificationType

The enumeration LcmOpNameForChangeNotificationType represents the name of the lifecycle operation that impacts the NS component and trigger an NS change notification. It shall comply with the provisions defined in Table 6.5.4.6-1.

Table 6.5.4.6-1: Enumeration LcmOpNameForChangeNotificationType

Enumeration value	Description
VNF_INSTANTIATE	Represents the "Instantiate VNF" LCM operation.
VNF_SCALE	Represents the "Scale VNF" LCM operation.
VNF_SCALE_TO_LEVEL	Represents the "Scale VNF to Level" LCM operation.
VNF_CHANGE_FLAVOUR	Represents the "Change VNF Flavour" LCM operation.
VNF_TERMINATE	Represents the "Terminate VNF" LCM operation.
VNF_HEAL	Represents the "Heal VNF" LCM operation.
VNF_OPERATE	Represents the "Operate VNF" LCM operation.
VNF_CHANGE_EXT_CONN	Represents the "Change external VNF connectivity" LCM operation.
VNF_MODIFY_INFO	Represents the "Modify VNF Information" LCM operation.
NS_INSTANTIATE	Represents the "Instantiate NS" LCM operation.
NS_SCALE	Represents the "Scale NS" LCM operation.
NS_UPDATE	Represents the "Update NS" LCM operation.
NS_TERMINATE	Represents the "Terminate NS" LCM operation.
NS_HEAL	Represents the "Heal NS" LCM operation.

6.5.4.7 Enumeration: LcmOpOccStatusForChangeNotificationType

The enumeration LcmOpOccStatusForChangeNotificationType represents the status of the lifecycle management operation occurrence that impacts the NS component and triggers an NS change notification. It shall comply with the provisions defined in Table 6.5.4.7-1.

Table 6.5.4.7-1: Enumeration LcmOpOccStatusForChangeNotificationType

Enumeration value	Description
START	The impact on the NS component is identified.
COMPLETED	The impact on the NS component stops and related lifecycle operation completes successfully.
PARTIALLY_COMPLETED	The impact on the NS component stops and related lifecycle operation partially completes. Inconsistency state may exist on the NS component.
FAILED	The impact on the NS component stops and related lifecycle operation fails. Inconsistency state may exist for the NS component.
ROLLED_BACK	The impact on the NS component stops and related lifecycle operation is rolled back.

6.5.4.8 Enumeration: OperationalStates

The enumeration OperationalStates shall comply with the provisions defined in Table 6.5.4.8-1.

Table 6.5.4.8-1: Enumeration OperationalStates

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

6.5.4.9 Enumeration: StopType

The enumeration StopType shall comply with the provisions defined in Table 6.5.4.9-1.

Table 6.5.4.9-1: Enumeration StopType

Enumeration value	Description
FORCEFUL	The VNFM will stop the VNF immediately after accepting the request.
GRACEFUL	The VNFM will first arrange to take the VNF out of service after accepting the request. Once that operation is successful or once the timer value specified in the "gracefulStopTimeout" attribute expires, the VNFM will stop the VNF.

6.5.4.10 Enumeration: CancelModeType

The enumeration CancelModeType defines the valid modes of cancelling a NS LCM operation occurrence. It shall comply with the provisions defined in Table 6.5.4.10-1.

Table 6.5.4.10-1: Enumeration CancelModeType

Enumeration value	Description
GRACEFUL	The NFVO shall not start any new VNF lifecycle management and resource management operation, and shall wait for the ongoing VNF lifecycle management and resource management operations in the underlying system, typically the VNFM and VIM, to finish execution or to time out. After that, the NFVO shall put the operation occurrence into the FAILED_TEMP state.
FORCEFUL	The NFVO shall not start any new VNF lifecycle management and resource management operation, shall cancel the ongoing VNF lifecycle management and resource management operations in the underlying system, typically the VNFM and VIM, and shall wait for the cancellation to finish or to time out. After that, the NFVO shall put the operation occurrence into the FAILED_TEMP state.

6.6 Handling of errors during NS lifecycle management operations

6.6.1 Basic concepts (informative)

6.6.1.1 Motivation

NS 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.

6.6.1.2 Failure resolution strategies: Retry, Rollback and Continue

Most transient errors are handled best with a retry mechanism. Retry might happen automatically at the point of failure within the same NS 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 system needs to decide the error handling strategy, either by a backward action or a forward action. By a backward action, it means the concerned error is not acceptable and permanent. Therefore, the system attempts to resolve the inconsistent state by requesting to roll back the changes made by the operation. By a forward action, it means the concerned error is acceptable and can be fixed later (typically after current operation). Therefore, the system decides to skip the concerned error and continues the operation, e.g. based on policy configuration. Given that, rollback and continue as error handling strategies are 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 either rollback or continue if the retry has failed.

Idempotent retry is meaningful and useful for all operation types. 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 InstantiateNs or scale out, while ill-suited for subtractive ones such as scale in or TerminateNs, or for HealNs. For some operations, continue is better suited if the concerned error is acceptable.

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 the OSS/BSS or human operator.

6.6.1.3 Error handling at NFVO and OSS/BSS

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

- Stop on first error:
    - Once the NFVO encounters an error, the normal execution of the NS LCM workflow is interrupted, and an error handling procedure is triggered (i.e. automatic retry, automatic rollback, automatic fail, escalate). See the paragraphs below for description of error handling procedures.
    - It is assumed that all NSs and all NFVOs support "stop on first error".

EXAMPLE 1: OSS/BSS is attempting to instantiate a NS with 10 VNFs. The first 8 VNFs are instantiated successfully, however, an error occurs when attempting to instantiate VNF #9. The NFVO 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 NFVO encounters an error, it is decided whether the execution of a part or all of the remaining steps of the NS LCM workflow is performed, or whether the execution is interrupted and an error handling procedure is triggered (i.e. 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 NSs and not all NFVOs support "best effort".

EXAMPLE 2: Same example as above. After the error occurs attempting to instantiate VNF #8, the NFVO continues by creating #9 and #10, and then chooses which error handling options it invokes.

The NFVO has the following error handling procedures to react to errors (see clause 6.6.1.2 for general elaboration regarding retry, rollback, and continue):

- Automatic Retry: The NFVO 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 NS LCM operation to be reported as successful. Failed retry is typically escalated.
- Automatic Rollback: The NFVO rolls back the NS to the state prior to starting the NS LCM operation without involving an external entity. Rollback can eventually succeed or can fail, preventing the NS from reaching that previous state. Successful rollback leads to the NS LCM operation to be reported as rolled back. Failed rollback is typically escalated.
- Automatic Continue: The NFVO skips the error and continue the NS LCM operation without involving an external entity. Continue can eventually succeed or fail. Successful continue leads to the NS LCM operation to be reported as partially completed. Failed continue is typically escalated or trying other error handling procedures like automatic rollback.
- Escalate: After failed automatic retry/retries, automatic rollback or automatic continue is typically not the first option in most situations, but the error is preferably reported to the OSS/BSS for further resolution. The same applies if no automatic error resolution was attempted by the NFVO, or if automatic rollback has failed or if automatic continue is not appropriate (e.g. based on policy configuration). This is done by sending a NS LCM operation occurrence notification.
- Unresolvable Error: The NFVO determines that the operation has failed and definitely cannot be recovered (e.g. if no retry, no continue, and no rollback is possible), and that escalating the error to the OSS/BSS will have no chance to lead to a resolution either. In this case, the NFVO would report that the operation has terminally failed. After that, other means of resolution can be attempted, such as the invocation of Heal NS, or manual procedures using the GUI of the NFVO or VIM to release stranded resources.

The OSS/BSS has the following error handling procedures to react to error reports from the NFVO:

- On-demand retry: After the NFVO has reported the error to the OSS/BSS, the OSS/BSS 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 NFVO by the OSS/BSS via the NS LCM interface.
- On-demand rollback: After the NFVO has reported the error to the OSS/BSS, and after the OSS/BSS or the human operator has decided to roll back the operation, the rollback of the operation is triggered towards the NFVO by the OSS/BSS via the NS LCM interface.
- On-demand continue: After the NFVO has reported the error to the OSS/BSS, and after the OSS/BSS or the human operator has decided to continue the operation, the continue of the operation is triggered towards the NFVO by the OSS/BSS via the NS LCM interface.

- Fail: After the NFVO has reported the error to the OSS/BSS, and after the OSS/BSS or the human operator has determined that neither on-demand retry nor on-demand rollback will fix the error, or on-demand continue is not appropriate, the NS LCM operation can be declared as terminally failed towards the NFVO by the OSS/BSS via the NS LCM interface. After that, other means of resolution can be attempted, such as the invocation of HealNs, or manual procedures using the GUI of the NFVO or VIM to release stranded resources.

6.6.2 States and state transitions of a NS lifecycle management operation occurrence

6.6.2.1 General

A NS 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 6.6.2.1-1. Transitions labelled with underlined text represent error handling operations; other transitions represent conditions.

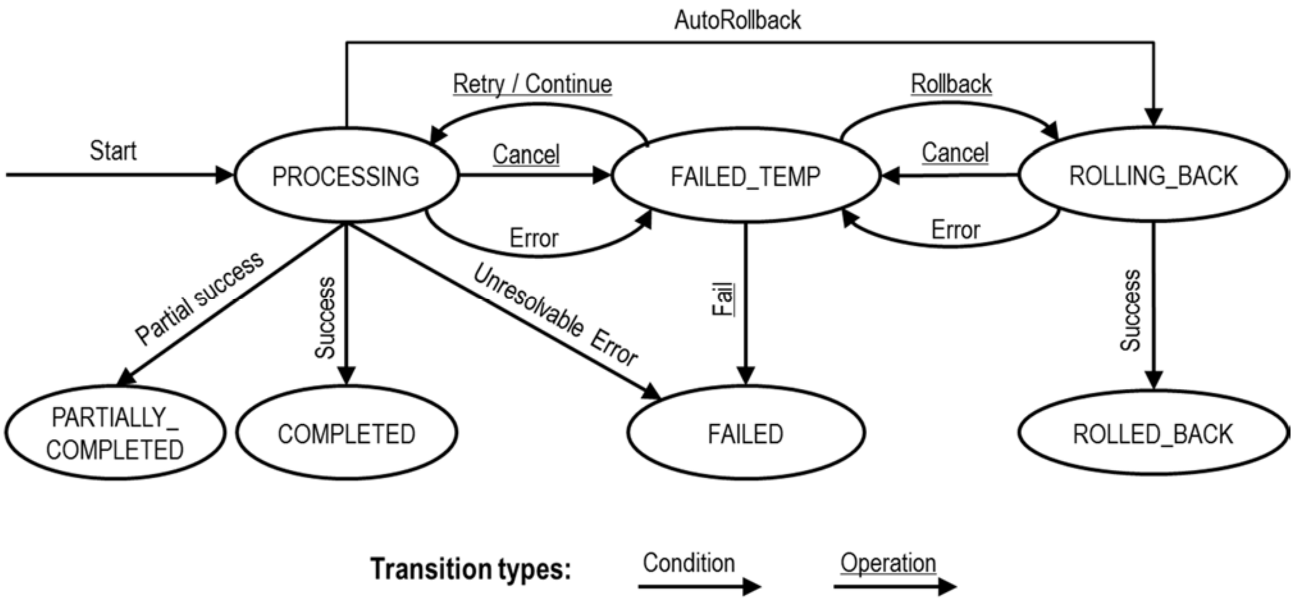

Figure 6.6.2.1-1: States of a NS lifecycle management operation occurrence

6.6.2.2 States of a NS lifecycle management operation occurrence

At each time, a NS 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 NS lifecycle management operation occurrence in a terminal state cannot change anymore).

PROCESSING: The NS LCM operation is currently in execution. This state has the following characteristics:

- This is the initial state for any NS operation.
- This is a transient state.
- This state may block other NS LCM operations from being executed on the same NS instance (up to NS and NFVO implementation).
- The operations "Retry", "Continue", "Fail", and "Rollback" shall not be permitted to be invoked for an operation that is in this state.

- All failures of procedures executed by the NFVO as part of the NS 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 NFVO knows that the NS instance can be brought into a consistent state by immediately rolling back the operation, the NS lifecycle management operation occurrence may transit directly into the "ROLLING_BACK" state ("AutoRollback").
  - If a failure occurs in the "PROCESSING" state from which the NFVO knows that it can neither be fixed by retrying nor be rolled back nor be skipped by continuing, the NS lifecycle management operation occurrence may transit directly into the "FAILED" state ("Unresolvable Error").
  - If a failure occurs in the "PROCESSING" state from which the NFVO knows that the failure is acceptable and continues the NS LCM operation till it finishes, the NS lifecycle management operation occurrence may transit directly into the "PARTIALLY_COMPLETED" state ("Partial success").
- 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 NS lifecycle management operation occurrence shall transit into the "FAILED_TEMP" state.

COMPLETED: The operation has completed successfully. This is a terminal state.

PARTIALLY COMPLETED: The operation has completed partially, i.e. with acceptable errors. This state has the following characteristics:

- This is a terminal state.
- Such an operation state is typically the result of an automatic continue operation inside the NFVO or an on-demand continue operation from a higher layer management entity (i.e. OSS/BSS) for a given error.
- The result of the NS LCM operation (the actual resource changes) can show an inconsistent state of the NS. Nevertheless, these changes shall be synchronized between the NFVO and OSS/BSS (by reporting them in the LCCN, and by allowing the OSS/BSS to obtain them on request) in order for other NS LCM operations (e.g. Heal, Terminate, Update) to be guaranteed to work on resources that are known to the OSS/BSS.

The fact that a LCM operation is in "PARTIALLY_COMPLETED" state shall not block other operations from execution on the NS instance by the NFVO. However, the NS instance may itself be in a state that disallows certain operations.

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.
- This state may block other NS LCM operations from being executed on the same NS instance (enforced by the NFVO, and up to NS and NFVO capabilities).
- Retry and/or rollback and/or continue and/or fail may be invoked for the operation.
- If the NS LCM operation is retried or continued, the NS lifecycle management operation occurrence shall transit into the "PROCESSING" state.
- If the NS LCM operation is rolled back, the NS lifecycle management operation occurrence shall transit into the "ROLLING_BACK" state.
- If the NS LCM operation is marked as "failed", the NS 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, rolled back, or continued, 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 (i.e. OSS/BSS) or its human operator that an operation in "FAILED_TEMP" state cannot be retried or rolled back or continued ("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, rolled back, nor continued ("Unresolvable Error").
- The result of the NS LCM operation (the actual resource changes) can show an inconsistent state of the NS. Nevertheless, these changes shall be synchronized between the NFVO and OSS/BSS (by reporting them in the LCCN, and by allowing the OSS/BSS to obtain them on request) in order for other NS LCM operations (e.g. Heal, Terminate) to be guaranteed to work on resources that are known to the OSS/BSS.
- The fact that a LCM operation is in "FAILED" state shall not block other operations from execution on the NS instance by the NFVO. However, the NS instance may itself be in a state that disallows certain operations.

ROLLED_BACK: The state of the NS 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.

ROLLING_BACK: The NS LCM operation is currently being rolled back. This state has the following characteristics:

- This is a transient state.
- This state may block other NS LCM operations from being executed on the same NS instance (up to NS and NFVO implementation).
- The operations "Retry", "Continue", 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 NS 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 NS lifecycle management operation occurrence shall transit into the "ROLLED_BACK" state.

In addition, the following provisions apply to NS lifecycle management operation occurrence notifications:

- The "start" notification (i.e. notificationStatus="START") shall be sent when the operation enters one of states "PROCESSING" and "ROLLING_BACK" from another state, indicating the state entered.
- The "result" notification (i.e. notificationStatus="RESULT") shall be sent when the NS 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 NS'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" or partial success state "PARTIALLY_COMPLETED", indicating the state entered and the changes to the NS's resources.

Such a notification scheme allows the OSS/BSS to keep in sync with changes to the NS's resources by an ongoing NS 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 NS's resources will be performed by the related NS lifecycle management operation occurrence, and the OSS/BSS 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, a query of the resource that represents the NSlifecycle operation occurrence can be used by the OSS/BSS to obtain the same information.

6.6.2.3 Error handling operations that change the state of a NS lifecycle operation

Retry: This operation retries a NS lifecycle operation. It has the following characteristics:

- Execution of "Retry" for an actual NS LCM operation on a particular NS may be supported, depending on characteristics of the NS and the NS LCM operation.
- The operation may be invoked via an interface, or the NFVO may invoke the operation per its own decision.

Rollback: This operation rolls back a NS lifecycle operation. It has the following characteristics:

- Execution of "Rollback" for an actual NS LCM operation on a particular NS may be supported, depending on characteristics of the NS and the NS LCM operation.
- The operation may be invoked via an interface, or the NFVO may invoke the operation per its own decision.

Continue: This operation continues a NS lifecycle operation. It has the following characteristics:

- Execution of "Continue" for an actual NS LCM operation on a particular NS may be supported, depending on characteristics of the NS and the NS LCM operation.
- The operation may be invoked via an interface, or the NFVO may invoke the operation per its own decision.

Fail: This operation transits the NS lifecycle operation occurrence into the terminal "FAILED" state. It has the following characteristics:

- Execution of "Fail" shall be supported for a LCM operation on a particular NS if at least one of following - Retry, Rollback, Continue, or Cancel - is supported for this operation.
- The operation may be invoked via an interface, or the NFVO may invoke the operation per its own decision.

Cancel: This operation cancels an ongoing NS lifecycle management operation, its Retry, Rollback, or Continue. It has the following characteristics:

- Execution of "Cancel" for an actual NS LCM operation on a particular NS may be supported, depending on characteristics of the NS and the NS 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 NFVO will not initiate any new operation towards the underlying systems, will wait until the currently executed operations finish or time out, and will then put the NS lifecycle management operation occurrence into the "FAILED_TEMP" state.
    - When executing the *forceful* "Cancel" operation, the NFVO 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 or time out, and will the put the NS lifecycle management operation occurrence into the "FAILED_TEMP" state.
- Executing "Cancel" can lead to inconsistencies between the information that the NFVO has about the state of the resources of the NS, and their actual state. The probability of such inconsistencies is bigger when using the *forceful* cancellation mode.

6.6.3 Detailed flows

6.6.3.1 Immediate failure

If the NS LCM operation fails immediately, i.e. it returns an HTTP error, then the operation has not started, and no "NS LCM operation occurrence resource" has been created. Also, a "start" lifecycle management operation occurrence notification has not been sent. The operation cannot be retried, but the same operation may be invoked again from the API. The NS instance is not changed by a synchronous failure, so no special error handling is required.

Figure 6.6.3.1-1 illustrates the flow.

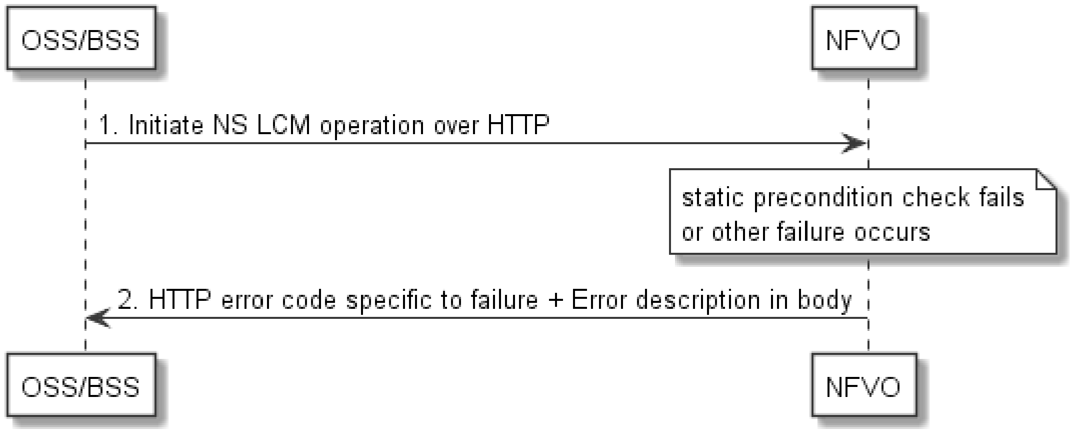
OSS/BSS
NFVO
1. Initiate NS LCM operation over HTTP
static precondition check fails
or other failure occurs
2. HTTP error code specific to failure + Error description in body
OSS/BSS
NFVO

Figure 6.6.3.1-1: Immediate failure of a NS LCM operation

6.6.3.2 Failure during actual NS LCM operation execution

After a failed resource management operation, automatic retry can be invoked by the NFVO itself. These invocations are not visible outside of the NFVO, as the NS 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 NS LCM operation is set into a temporary failure state, and the OSS/BSS 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 6.6.3.2-1 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.

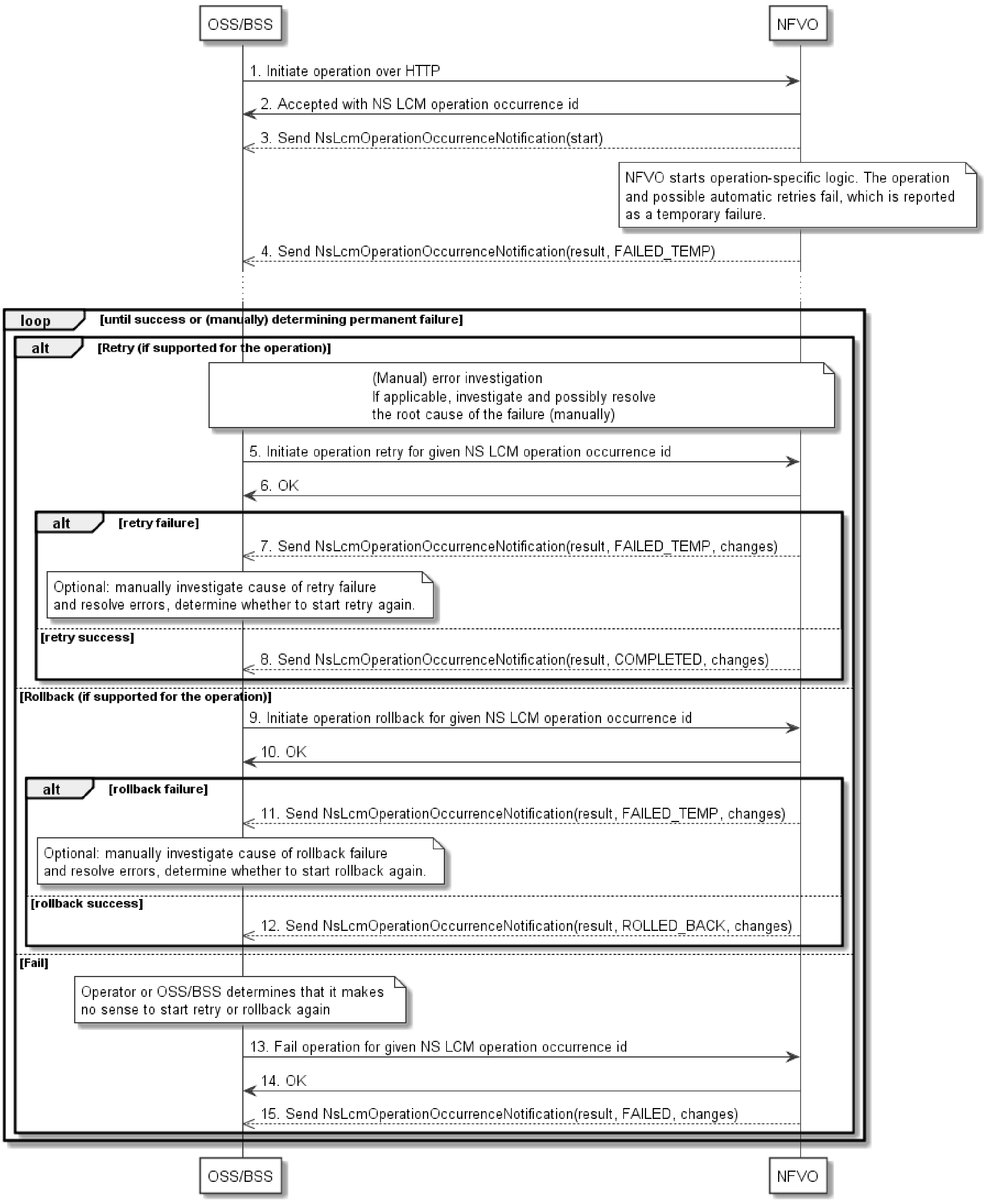
OSS/BSS
NFVO
1. Initiate operation over HTTP
2. Accepted with NS LCM operation occurrence id
3. Send NsLcmOperationOccurrenceNotification(start)
NFVO starts operation-specific logic. The operation and possible automatic retries fail, which is reported as a temporary failure.
4. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP)
loop [until success or (manually) determining permanent failure]
alt [Retry (if supported for the operation)]
(Manual) error investigation If applicable, investigate and possibly resolve the root cause of the failure (manually)
5. Initiate operation retry for given NS LCM operation occurrence id
6. OK
alt [retry failure]
7. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Optional: manually investigate cause of retry failure and resolve errors, determine whether to start retry again.
[retry success]
8. Send NsLcmOperationOccurrenceNotification(result, COMPLETED, changes)
[Rollback (if supported for the operation)]
9. Initiate operation rollback for given NS LCM operation occurrence id
10. OK
alt [rollback failure]
11. Send NsLcmOperationOccurrenceNotification(result, FAILED_TEMP, changes)
Optional: manually investigate cause of rollback failure and resolve errors, determine whether to start rollback again.
[rollback success]
12. Send NsLcmOperationOccurrenceNotification(result, ROLLED_BACK, changes)
[Fail]
Operator or OSS/BSS determines that it makes no sense to start retry or rollback again
13. Fail operation for given NS LCM operation occurrence id
14. OK
15. Send NsLcmOperationOccurrenceNotification(result, FAILED, changes)
OSS/BSS
NFVO

Figure 6.6.3.2-1: Handling failures during the actual execution of a NS LCM operation

6.6.3.3 LCM operation cancellation

The cancellation of a NS LCM operation that is in PROCESSING or ROLLING_BACK state is handled like any other error that leads to stopping the execution of the NS LCM workflow before it can be successfully completed. The NS 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.

7 NS Performance Management interface

7.1 Description

This interface allows providing performance management (measurement results collection and notifications) related to NSs. Performance information on a given NS instance is sent by the NFVO to the OSS/BSS. 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 NsPerformanceInformationAvailableNotification.

The operations provided through this interface are:

- Create PM Job
- Query PM Job
- Delete PM Job
- Create Threshold
- Query Threshold
- Delete Threshold
- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

7.1a API version

For the NS performance management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 1, and the PATCH version number shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.4.1 of the present document.

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 "nspm" shall be used to represent {apiName}. 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 performance management API.

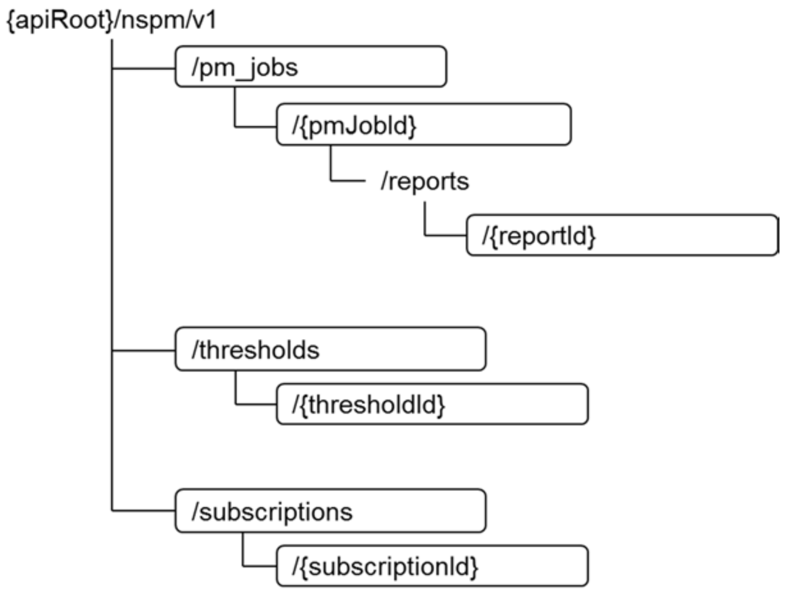

Figure 7.2-1: Resource URI structure of the NS Performance Management interface

Table 7.2-1 lists the individual resources defined, and the applicable HTTP methods. The NFVO shall support responding to requests for all HTTP methods on the resources in Table 7.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 7.2-1: Resources and methods overview of the NS Performance Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
PM jobs	/pm_jobs	POST	M	Create a PM job
		GET	M	Query PM jobs
Individual PM job	/pm_jobs/{pmJobId}	GET	M	Read an individual PM job
		DELETE	M	Delete a PM job
Individual performance report	/pm_jobs/{pmJobId}/reports/{reportId}	GET	M	Read an individual performance report
Thresholds	/thresholds	POST	M	Create a threshold
		GET	M	Query thresholds
Individual threshold	/thresholds/{thresholdId}	GET	M	Read a single threshold
		DELETE	M	Delete a threshold
Subscriptions	/subscriptions	POST	M	Subscribe to PM notifications
		GET	M	Query PM related subscriptions
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read a single PM related subscription
		DELETE	M	Terminate a subscription
Notification endpoint	(client-defined)	POST	See note	Notify about PM related events. See note
		GET	See note	Test the notification endpoint. See note
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the OSS/BSS. If the OSS/BSS supports invoking the POST method on the "Subscriptions" resource towards the NFVO, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

7.3 Sequence diagrams (informative)

7.3.1 Flow of creating a PM job

This clause describes a sequence for creating a performance management jobs.

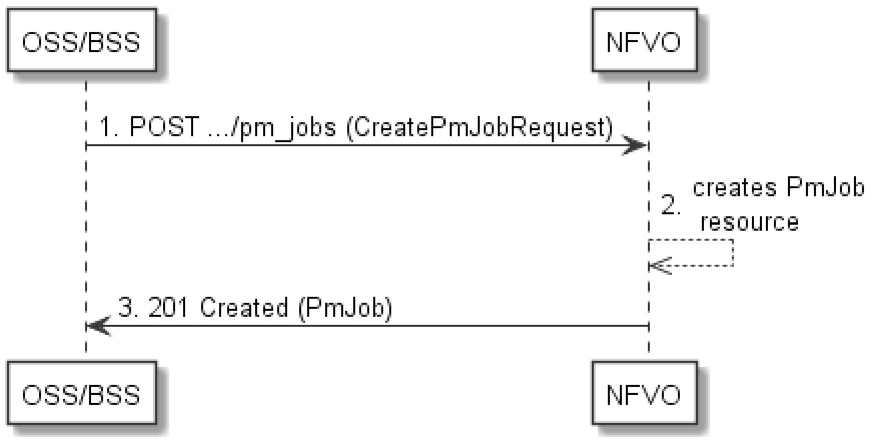
OSS/BSS
NFVO
1. POST .../pm_jobs (CreatePmJobRequest)
2. creates PmJob resource
3. 201 Created (PmJob)
OSS/BSS
NFVO

Figure 7.3.1-1: Flow of PM job creation

PM job creation, as illustrated in Figure 7.3.1-1, consists of the following steps:

1) If the OSS/BSS 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 NFVO creates a PM job instance.

3) The NFVO returns a "201 Created" response to the OSS/BSS, 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.

7.3.2 Flow of querying/reading PM jobs

This clause describes a sequence for querying/reading performance management jobs.

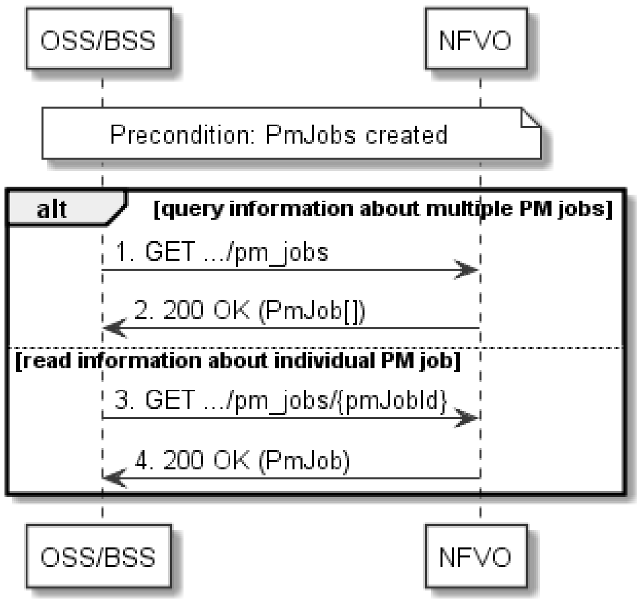
OSS/BSS
NFVO
Precondition: PmJobs created
alt
[query information about multiple PM jobs]
1. GET .../pm_jobs
2. 200 OK (PmJob[])
[read information about individual PM job]
3. GET .../pm_jobs/{pmJobId}
4. 200 OK (PmJob)
OSS/BSS
NFVO

Figure 7.3.2-1: Flow of PM jobs query/read

PM jobs query/read, as illustrated in Figure 7.3.2-1, consists of the following steps:

1) If the OSS/BSS intends to query all PM jobs, it sends a GET request to the "PM jobs" resource.

2) The NFVO returns a "200 OK" response to the OSS/BSS, and includes zero or more data structures of type "PmJob" in the payload body.

3) If the OSS/BSS 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 NFVO returns a "200 OK" response to the OSS/BSS, 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.

7.3.3 Flow of deleting a PM job

This clause describes a sequence for deleting a performance management jobs.

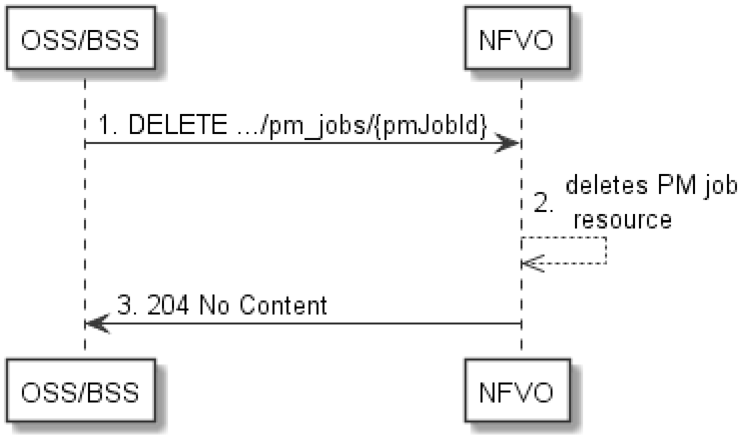

Figure 7.3.3-1: Flow of PM job deletion

PM job deletion, as illustrated in Figure 7.3.3-1, consists of the following steps:

1) If the OSS/BSS 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 NFVO deletes the PM Job instance.

3) The NFVO returns a response with a "204 No Content" response code and an empty payload body to the OSS/BSS.

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

7.3.4 Flow of obtaining performance reports

This clause describes a sequence for obtaining performance reports.

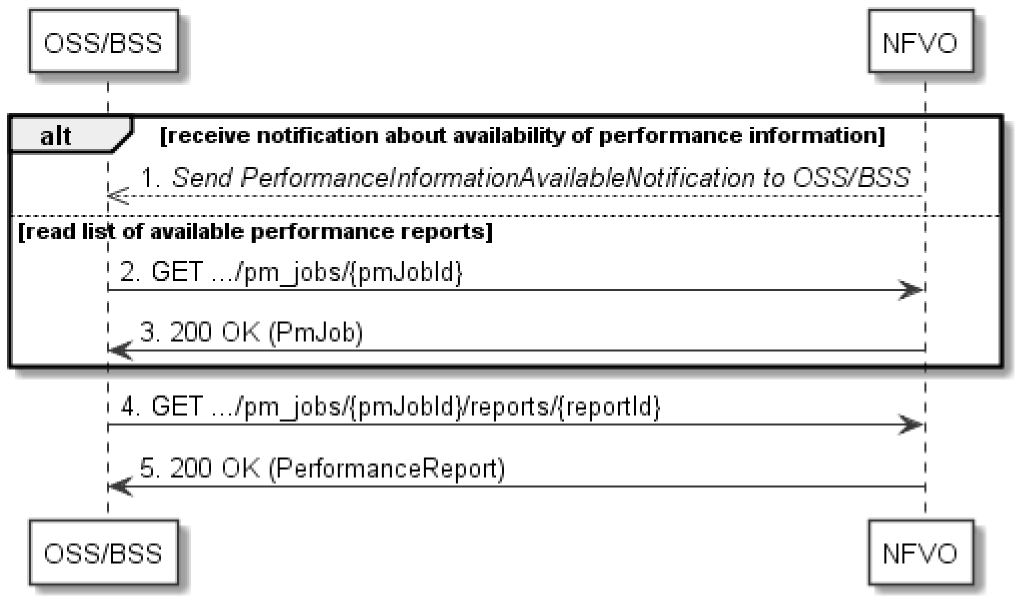
OSS/BSS
NFVO
alt
[receive notification about availability of performance information]
1. Send PerformanceInformationAvailableNotification to OSS/BSS
[read list of available performance reports]
2. GET .../pm_jobs/{pmJobId}
3. 200 OK (PmJob)
4. GET .../pm_jobs/{pmJobId}/reports/{reportId}
5. 200 OK (PerformanceReport)
OSS/BSS
NFVO

Figure 7.3.4-1: Flow of obtaining performance reports

Obtaining a performance report, as illustrated in Figure 7.3.4-1, consists of the following steps:

1) The NFVO sends to the OSS/BSS a PerformanceInformationAvailableNotification (see clause 7.3.9) that indicates the availability of a new performance report, including a link from which the report can be obtained.

2) Alternatively, the OSS/BSS 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 NFVO returns a "200 OK" response to the OSS/BSS, and includes a data structure of type "PmJob" in the payload body.

4) The OSS/BSS sends to the NFVO a GET request to the URI obtained either in step (1) or step (3), in order to read a performance report resource.

5) The NFVO returns a "200 OK" response to the OSS/BSS, and includes a data structure of type "PerformanceReport" in the payload body.

7.3.5 Flow of creating a threshold

This clause describes a sequence for creating a performance management threshold.

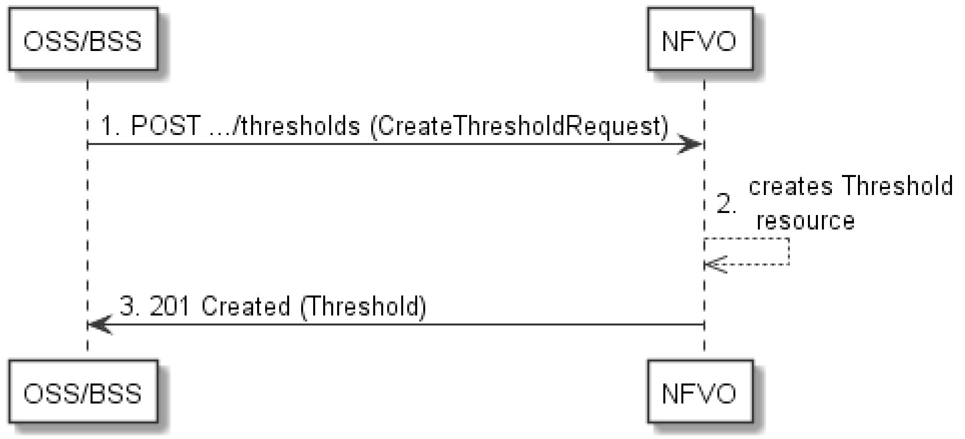
OSS/BSS
NFVO
1. POST .../thresholds (CreateThresholdRequest)
2. creates Threshold resource
3. 201 Created (Threshold)
OSS/BSS
NFVO

Figure 7.3.5-1: Flow of threshold creation

Threshold creation, as illustrated in Figure 7.3.5-1, consists of the following steps:

1) If the OSS/BSS 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 NFVO creates a threshold instance.

3) The NFVO returns a "201 Created" response to the OSS/BSS, 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.

7.3.6 Flow of querying/reading thresholds

This clause describes a sequence for querying/reading performance management thresholds.

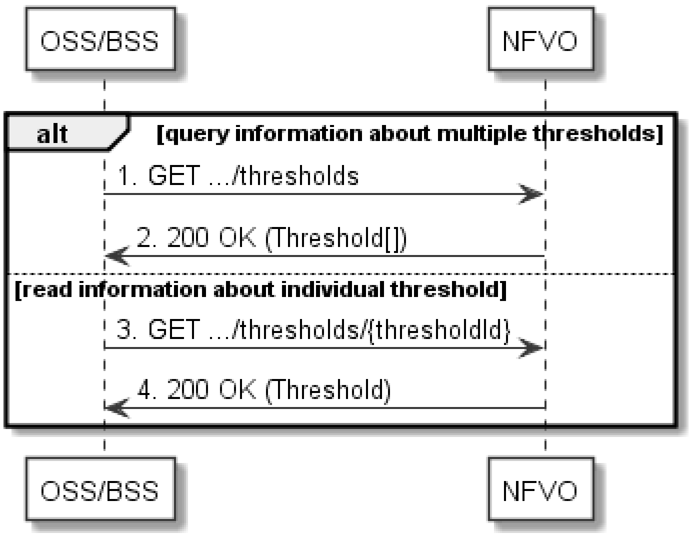
OSS/BSS
NFVO
alt
[query information about multiple thresholds]
1. GET .../thresholds
2. 200 OK (Threshold[])
[read information about individual threshold]
3. GET .../thresholds/{thresholdId}
4. 200 OK (Threshold)
OSS/BSS
NFVO

Figure 7.3.6-1: Flow of thresholds query/read

Threshold query/read, as illustrated in Figure 7.3.6-1, consists of the following steps:

1) If the OSS/BSS intends to query all thresholds, it sends a GET request to the "Thresholds" resource.

2) The NFVO returns a "200 OK" response to the OSS/BSS, and includes zero or more data structures of type "Threshold" in the payload body.

3) If the OSS/BSS intends to read information about a particular threshold, it sends a GET request to the "Individual threshold" resource with the appropriate threshold identifier in its resource URI.

4) The NFVO returns a "200 OK" response to the OSS/BSS, 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.

7.3.7 Flow of deleting thresholds

This clause describes a sequence for deleting performance management thresholds.

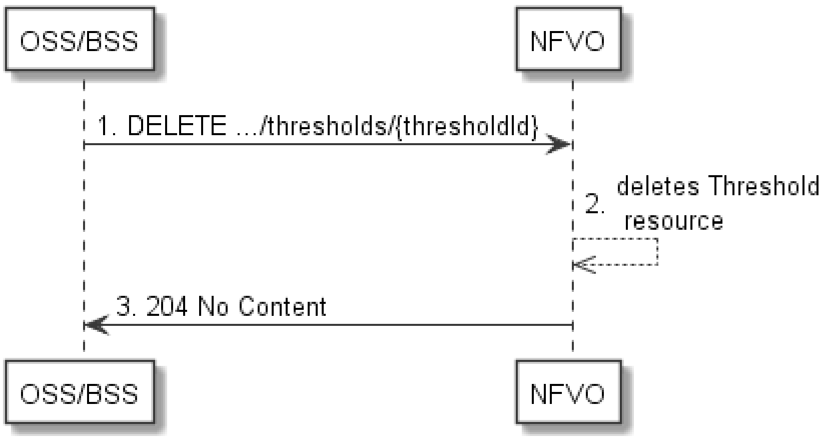
OSS/BSS
NFVO
1. DELETE .../thresholds/{thresholdId}
2. deletes Threshold resource
3. 204 No Content
OSS/BSS
NFVO

Figure 7.3.7-1: Flow of threshold deletion

Threshold deletion, as illustrated in Figure 7.3.7-1, consists of the following steps:

1) If the OSS/BSS 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 NFVO returns a "204 No Content" response code to the NFVO. The response body shall be empty.

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

7.3.8 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to NS performance management.

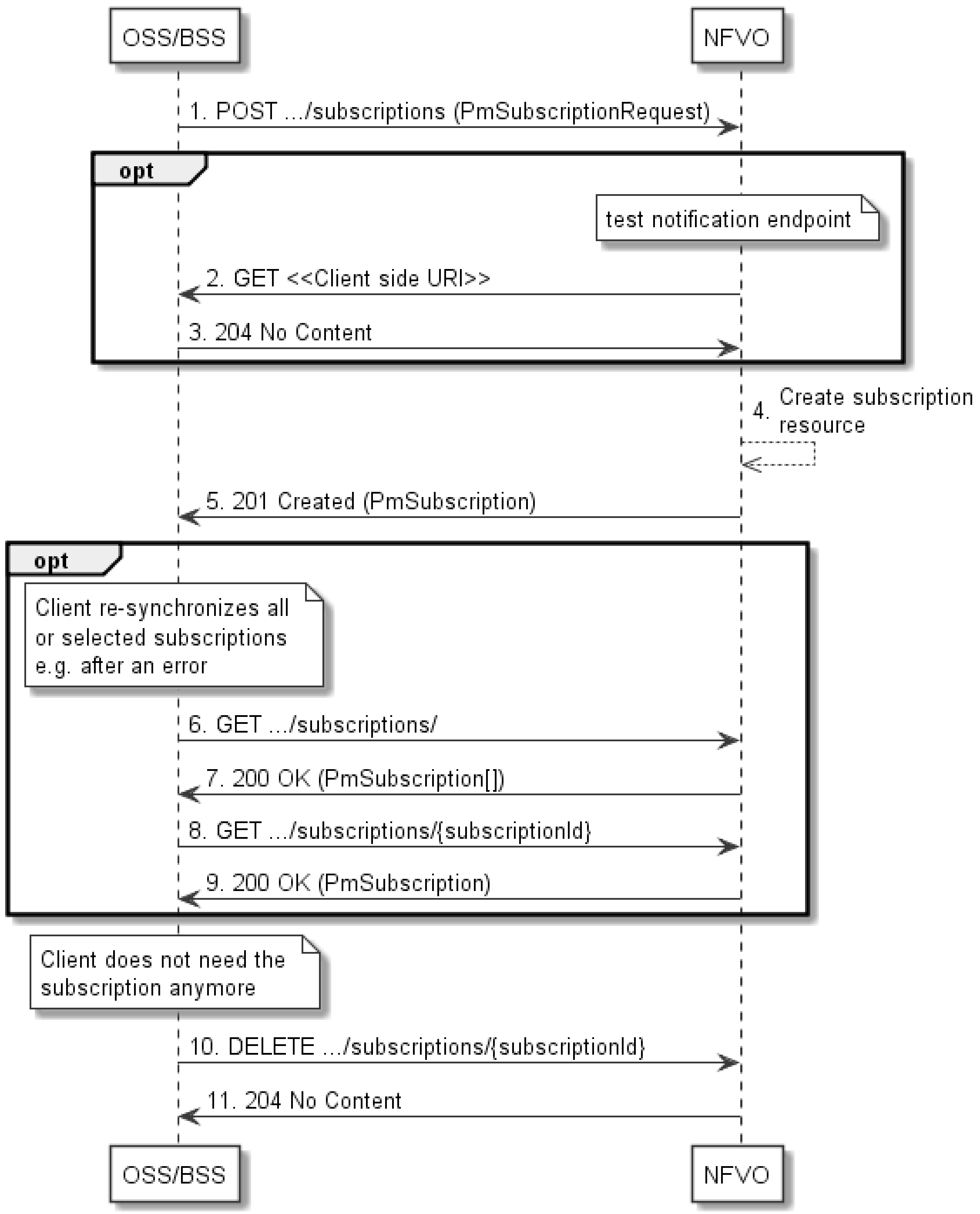
OSS/BSS
NFVO
1. POST .../subscriptions (PmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (PmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (PmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (PmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
OSS/BSS
NFVO

Figure 7.3.8-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in Figure 7.3.8-1:

1) The OSS/BSS 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 NFVO will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the OSS/BSS as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

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

4) The NFVO creates a new subscription to notifications related to NS performance management, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "PmSubscription," representing the subscription resource just created by the NFVO, 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 OSS/BSS may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the OSS/BSS.

8) Optionally, for example when trying to recover from an error situation, the OSS/BSS may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

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

10) When the OSS/BSS 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 NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

7.3.9 Flow of sending notifications

This clause describes the procedure for sending notifications related to NS performance management.

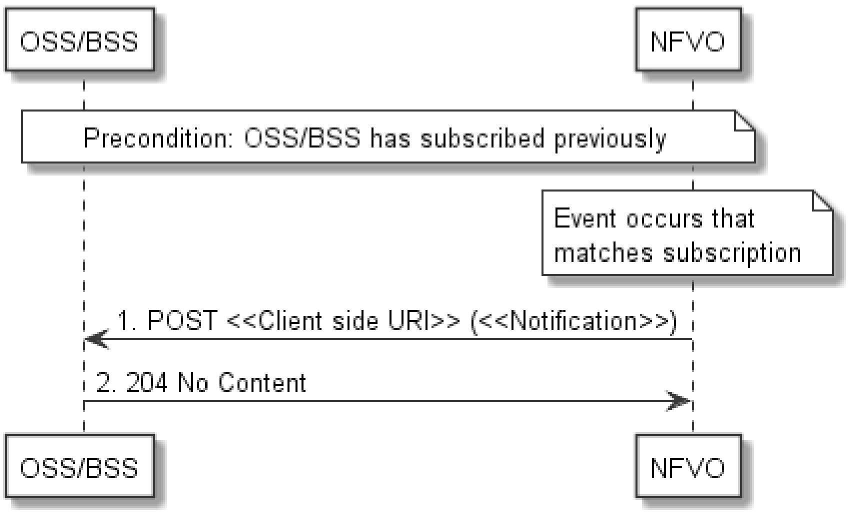
OSS/BSS
NFVO
Precondition: OSS/BSS has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
OSS/BSS
NFVO

Figure 7.3.9-1: Flow of sending notifications

Precondition: The OSS/BSS has subscribed previously for notifications related to NS performance management.

The procedure consists of the following steps as illustrated in Figure 7.3.9-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the OSS/BSS 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 OSS/BSS acknowledges the successful delivery of the notification by returning a "204 No Content" response.

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

7.4 Resources

7.4.1 Introduction

This clause defines all the resources and methods provided by the performance management API.

7.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the NS performance management interface.

7.4.2 Resource: PM jobs

7.4.2.1 Description

This resource represents PM jobs. The client can use this resource to create and query PM jobs.

7.4.2.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/pm_jobs

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

The POST method creates a PM job.

This method shall follow the provisions specified in the Tables 7.4.2.3.1-1 and 7.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.2.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	CreatePmJobRequest	1	PM job creation request	
Response body	Data type	Cardinality	Response Codes	Description
	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 7.5.2.7. 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.

7.4.2.3.2 GET

The client can use this method to retrieve information about PM jobs.

This method shall follow the provisions specified in the Tables 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 expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the PmJob and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
include	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO 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 NFVO 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.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

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**
	PmJob	0..N	200 OK	Information about zero or more PM jobs was queried successfully. The response body shall contain in an array the representations of zero or more PM jobs, as defined in clause 7.5.2.7. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

7.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3 Resource: Individual PM job

7.4.3.1 Description

This resource represents an individual PM job. The client can use this resource to delete and read the underlying PM job.

7.4.3.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/pm_jobs/{pmJobId}

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.
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.	

7.4.3.3 Resource methods

7.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3.3.2 GET

The client can use this method for reading an individual PM job.

This method shall follow the provisions specified in the Tables 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
	PmJob	1	200 OK	Information about an individual PM job was read successfully. The response body shall contain a representation of the PM job resource, as defined in clause 7.5.2.7.
	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.

7.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.3.3.5 DELETE

This method terminates an individual PM job.

This method shall follow the provisions specified in the Tables 7.4.3.3.5-1 and 7.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.3.3.5-2: 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.

7.4.4 Resource: Individual performance report

7.4.4.1 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 7.5.2.5) 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.

7.4.4.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/pm_jobs/{pmJobId}/reports/{reportId}

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.
pmJobId	Identifier of the PM job.
reportId	Identifier of the performance report.

7.4.4.3 Resource methods

7.4.4.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4.3.2 GET

The client can use this method for reading an individual performance report.

This method shall follow the provisions specified in the Tables 7.4.4.3.2-1 and 7.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.4.3.2-2: 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**
	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 7.5.2.10.
	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.

7.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5 Resource: Thresholds

7.4.5.1 Description

This resource represents thresholds. The client can use this resource to create and query thresholds.

7.4.5.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/thresholds

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 can be used by the client to create a threshold.

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.

Table 7.4.5.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
None supported		

Table 7.4.5.3.1-2: Details of the POST request/response on this resource

Request body	Data type	Cardinality	Description	
	CreateThresholdRequest	1	Request parameters to create a threshold resource.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	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 7.5.2.9. 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.

7.4.5.3.2 GET

The client can use this method to query information about thresholds.

This method shall follow the provisions specified in the Tables 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	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the Thresholds data type and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

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 7.4.5.3.2-2: 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 in an array the representations of zero or more thresholds, as defined in clause 7.5.2.9. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

7.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6 Resource: Individual threshold

7.4.6.1 Description

This resource represents an individual threshold.

7.4.6.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/thresholds/{thresholdId}

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.
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.	

7.4.6.3 Resource methods

7.4.6.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6.3.2 GET

The client can use this method for reading an individual threshold.

This method shall follow the provisions specified in the Tables 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**
	Threshold	1	200 OK	Information about an individual threshold was read successfully. The response body shall contain a representation of the threshold, as defined in clause 7.5.2.9.
	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.

7.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.6.3.5 DELETE

This method allows to delete a threshold.

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

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 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.

7.4.7 Resource: Subscriptions

7.4.7.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to NS performance management and to query its subscriptions.

7.4.7.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/subscriptions

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
apiRoot	See clause 4.2

7.4.7.3 Resource methods

7.4.7.3.1 POST

The POST method creates a new subscription.

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.

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 OSS, and might make sense only in very rare use cases. Consequently, the NFVO 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 7.4.7.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.7.3.1-2: 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 7.5.2.3. 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 exits and the policy of the NFVO 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.

7.4.7.3.2 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 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
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the PmSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 7.4.7.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
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 in an array the representations of all active subscriptions of the functional block that invokes the method, i.e. zero or more representations of PM subscriptions, as defined in clause 7.5.2.3. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

7.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.8 Resource: Individual subscription

7.4.8.1 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 NS performance management.

7.4.8.2 Resource definition

The resource URI is:

{apiRoot}/nspm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in Table 7.4.8.2-1.

Table 7.4.8.2-1: 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.	

7.4.8.3 Resource methods

7.4.8.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.8.3.2 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 7.4.8.3.2-1 and 7.4.8.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.8.3.2-2: 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 7.5.2.3.
	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.

7.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.8.3.5 DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the Tables 7.4.8.3.5-1 and 7.4.8.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.8.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.8.3.5-2: Details of the DELETE request/response on this resource

	Data type	Cardinality	Description	
Request body	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.

7.4.9 Resource: Notification endpoint

7.4.9.1 Description

This resource represents a notification endpoint for NS 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.

7.4.9.2 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 7.4.9.2-1.

Table 7.4.9.2-1: Resource URI variables for this resource

Name	Definition
n/a	

7.4.9.3 Resource methods

7.4.9.3.1 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 7.4.9.3.1-1 and 7.4.9.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.9.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.9.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	PerformanceInformationAvailableNotification	1	Notification about performance information availability	
	ThresholdCrossedNotification	1	Notification about threshold crossing	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	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.

7.4.9.3.2 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 7.4.9.3.2-1 and 7.4.9.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 7.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 7.4.9.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	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.

7.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.4.9.3.5 DELETE

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

7.5 Data Model

7.5.1 Introduction

This clause defines the request and response data structures of the NS Performance Management interface. If a request or response contains attributes not defined in the present document, 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 resource representations and notifications.

7.5.2.2 Type: PmSubscriptionRequest

This type represents a subscription request. It shall comply with the provisions defined in Table 7.5.2.2-1.

Table 7.5.2.2-1: 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.

7.5.2.3 Type: PmSubscription

This type represents a subscription. It shall comply with the provisions defined in Table 7.5.2.3-1.

Table 7.5.2.3-1: 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.

7.5.2.4 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 7.5.2.4-1.

Table 7.5.2.4-1: 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 that identifies a NS instance.
performanceMetric	String	1	Performance metric associated with the threshold. This attribute shall contain the related "Measurement Name" value as defined in clause 7.2 of ETSI GS NFV-IFA 027 [31].
performanceValue	(any type)	1	Value of the metric that resulted in threshold crossing. The type of this attribute shall correspond to the related "Measurement Unit" as defined in clause 7.2 of ETSI GS NFV-IFA 027 [31].
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.
>objectInstance	NotificationLink	0..1	Link to the resource representing the NS instance to which the notified change applies. Shall be present if the NS instance information is accessible as a resource.
>threshold	NotificationLink	1	Link to the resource that represents the threshold that was crossed.

7.5.2.5 Type: PerformanceInformationAvailableNotification

This notification informs the receiver that performance information is available. It shall comply with the provisions defined in Table 7.5.2.5-1.

NOTE: The timing of sending this notification is determined by the capability of the producing entity to evaluate the threshold crossing condition.

Table 7.5.2.5-1: 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 that identifies a NS instance.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.
>objectInstance	NotificationLink	0..1	Link to the resource representing the NS instance to which the notified change applies. Shall be present if the NS instance information is accessible as a resource.
>pmJob	NotificationLink	1	Link to the resource that represents the PM job for which performance information is available.
>performanceReport	NotificationLink	1	Link from which the available performance information of data type "PerformanceReport" (see clause 7.5.2.10) can be obtained. This link should point to an "Individual performance report" resource as defined in clause 7.4.4.

7.5.2.6 Type: CreatePmJobRequest

This type represents a request to create a PM job. It shall comply with the provisions defined in Table 7.5.2.6-1.

Table 7.5.2.6-1: Definition of the CreatePmJobRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceIds	Identifier	1..N	Identifiers of the NS instances for which performance information is requested to be collected.
criteria	PmJobCriteria	1	Criteria of the collection of performance information.

7.5.2.7 Type: PmJob

This type represents a PM job. It shall comply with the provisions defined in Table 7.5.2.7-1.

Table 7.5.2.7-1: 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 NS instances for which performance information is collected.
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 NS instances for which performance information is collected. Shall be present if the NS instance information is accessible as a resource.

7.5.2.8 Type: CreateThresholdRequest

This type represents a request to create a threshold. It shall comply with the provisions defined in Table 7.5.2.8-1.

Table 7.5.2.8-1: Definition of the CreateThresholdRequest data type

Attribute name	Data type	Cardinality	Description
objectInstanceId	Identifier	1	Identifier of the NS instance associated with this threshold.
criteria	ThresholdCriteria	1	Criteria that define this threshold.

7.5.2.9 Type: Threshold

This type represents a threshold. It shall comply with the provisions defined in Table 7.5.2.9-1.

Table 7.5.2.9-1: 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 NS instance associated with the threshold.
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 NS instance for which performance information is collected. Shall be present if the NS instance information is accessible as a resource.

7.5.2.10 Type: PerformanceReport

This type defines the format of a performance report provided by the NFVO to the OSS/BSS as a result of collecting performance information as part of a PM job. The type shall comply with the provisions defined in Table 7.5.2.10-1.

Table 7.5.2.10-1: 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. NS instance), but can include multiple collected values.
>objectType	String	1	Defines the object type for which performance information is reported (i.e. NS type). The string value shall be set to the nsdId of the NS instance to which the performance information relates.
>objectInstanceId	Identifier	1	The object instance for which the performance metric is reported. The object instances for this information element will be NS instances.
>performanceMetric	String	1	Name of the metric collected.
>performanceValue	Structure (inlined)	1..N	List of performance values with associated timestamp.
>>timeStamp	DateTime	1	Time stamp indicating when the data was collected.
>>performanceValue	(any type)	1	Value of the metric collected. See note.
NOTE: The type of the "performanceValue" attribute (i.e. scalar, structure (Object in JSON), or array (of scalars, arrays or structures / Objects)) is outside the scope of the present document.			

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: 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 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 PmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
nsInstanceSubscriptionFilter	NsfInstanceSubscriptionFilter	0..1	Filter criteria to select NS 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.			

7.5.3.3 Type: PmJobCriteria

This type represents collection criteria for PM jobs. It shall comply with the provisions defined in Table 7.5.3.3-1.

Table 7.5.3.3-1: 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. 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 notes 1 and 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 notes 1 and 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.
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.			

7.5.3.4 Type: ThresholdCriteria

This type represents criteria that define a threshold. It shall comply with the provisions defined in Table 7.5.3.4-1.

Table 7.5.3.4-1: Definition of the ThresholdCriteria data type

Attribute name	Data type	Cardinality	Description
performanceMetric	String	1	Defines the performance metric associated with the threshold.
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.

Attribute name	Data type	Cardinality	Description
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).			

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 4.4.2.

7.5.4.3 Enumeration: CrossingDirectionType

The enumeration CrossingDirectionType shall comply with the provisions defined in Table 7.5.4.3-1.

Table 7.5.4.3-1: Enumeration CrossingDirectionType

Enumeration value	Description
UP	The threshold was crossed in upward direction.
DOWN	The threshold was crossed in downward direction.

8 NS Fault Management interface

8.1 Description

This interface allows the OSS/BSS to subscribe to notifications regarding NS alarms provided by the NFVO. An alarm on a given NS results from either a collected virtualised resource fault impacting the connectivity of the NS instance or a VNF alarm, resulting from a virtualised resource alarm, issued by the VNFM for a VNF that is part of this NS instance.

The operations provided through this interface are:

- Get Alarm List
- Acknowledge Alarm

- Subscribe
- Query Subscription Information
- Terminate Subscription
- Notify

8.1a API version

For the NS fault management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 1, and the PATCH version number shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.4.1 of the present document.

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 "nsfm" shall be used to represent {apiName}. 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 NS fault management interface.

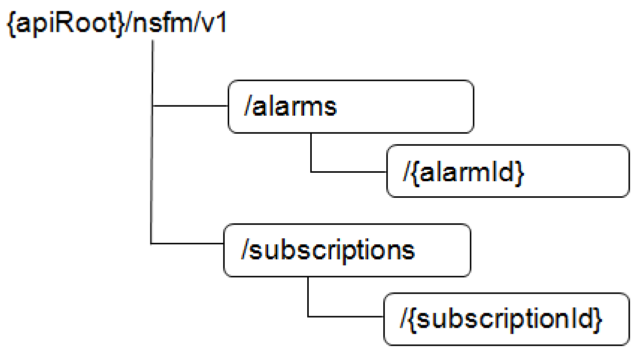

Figure 8.2-1: Resource URI structure of the NS Fault Management interface

Table 8.2-1 lists the individual resources defined, and the applicable HTTP methods. The NFVO shall support responding to requests for all HTTP methods on the resources in Table 8.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 8.2-1: Resources and methods overview of the NS Fault Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
Alarms	/alarms	GET	M	Query alarms related to NS instances.
Individual alarm	/alarms/{alarmId}	GET	M	Read individual alarm.
		PATCH	M	Acknowledge individual alarm.
Subscriptions	/subscriptions	POST	M	Subscribe to alarms related to NSs.
		GET	M	Query multiple subscriptions.
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription.
		DELETE	M	Terminate a subscription.
Notification endpoint	(client-provided)	POST	See note	Notify about NS alarms. See note.
		GET	See note	Test the notification endpoint. See note.
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the OSS/BSS. If the OSS/BSS supports invoking the POST method on the "Subscriptions" resource towards the NFVO, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

8.3 Sequence diagrams (informative)

8.3.1 Flow of the Get Alarm List operation

This clause describes a sequence flow for querying one or multiple alarms.

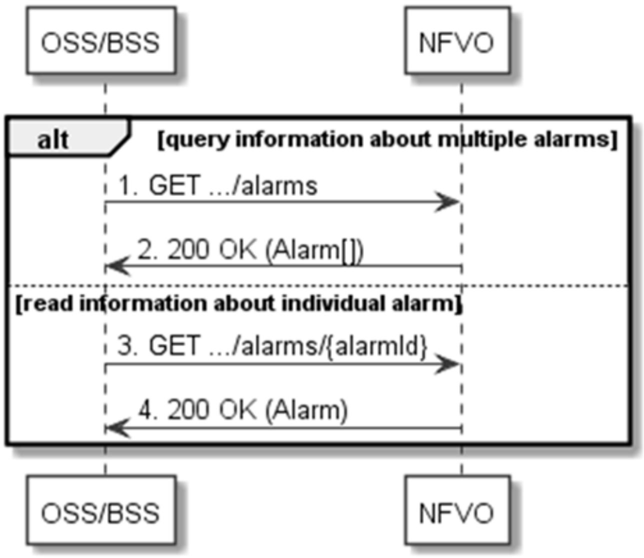
OSS/BSS
NFVO
alt
[query information about multiple alarms]
1. GET .../alarms
2. 200 OK (Alarm[])
[read information about individual alarm]
3. GET .../alarms/{alarmId}
4. 200 OK (Alarm)
OSS/BSS
NFVO

Figure 8.3.1-1: Flow of alarm query/read

Alarm query, as illustrated in Figure 8.3.1-1, consists of the following steps:

1) If the OSS/BSS intends to query all alarms, it sends a GET request to the "Alarms " resource.

2) The NFVO returns a "200 OK" response to the OSS/BSS, and includes zero or more data structures of type "Alarm" in the payload body.

3) If the OSS/BSS 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 NFVO returns a "200 OK" response to the OSS/BSS, 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.

8.3.2 Flow of acknowledging alarm

This clause describes the procedure to acknowledge an individual alarm.

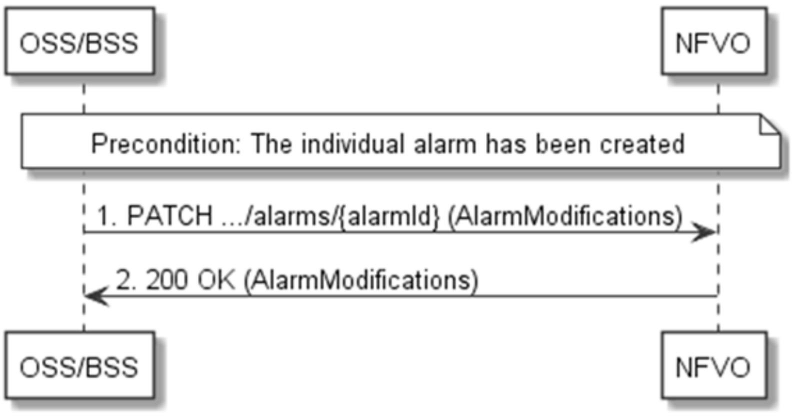
OSS/BSS
NFVO
Precondition: The individual alarm has been created
1. PATCH .../alarms/{alarmId} (AlarmModifications)
2. 200 OK (AlarmModifications)
OSS/BSS
NFVO

Figure 8.3.2-1: Flow of acknowledging alarm

Precondition: The resource representing the individual alarm has been created.

Acknowledge alarm, as illustrated in Figure 8.3.2-1, consists of the following steps:

1) The OSS/BSS sends a PATCH request to the individual alarm.

2) The NFVO returns a "200 OK" response to the OSS/BSS, 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.

8.3.3 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to NS fault management.

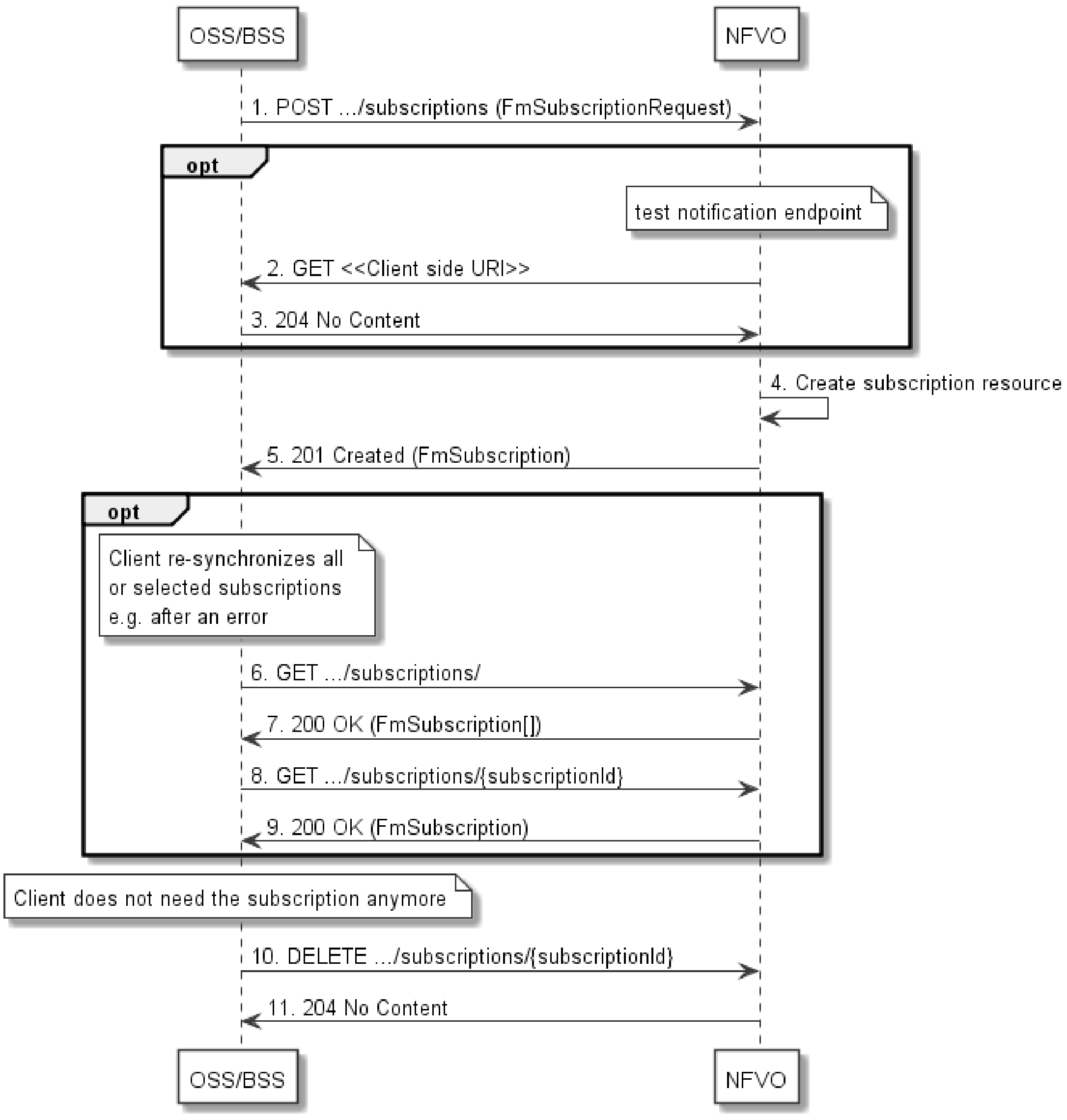
OSS/BSS
NFVO
1. POST .../subscriptions (FmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (FmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (FmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (FmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
OSS/BSS
NFVO

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 OSS/BSS 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 NFVO will subsequently send notifications about events that match the filter.

2) Optionally, to test the notification endpoint that was registered by the OSS/BSS as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

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

4) The NFVO creates a new subscription for notifications related to NS fault management, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "FmSubscription," representing the subscription resource just created by the NFVO, 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 OSS/BSS may query information about its subscriptions by sending a GET request to the "Subscriptions" resource.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the NFVO.

8) Optionally, for example when trying to recover from an error situation, the OSS/BSS may read information about a particular subscription by sending a GET request to the resource representing that individual subscription.

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

10) When the OSS/BSS 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 NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO 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 related to NS fault management.

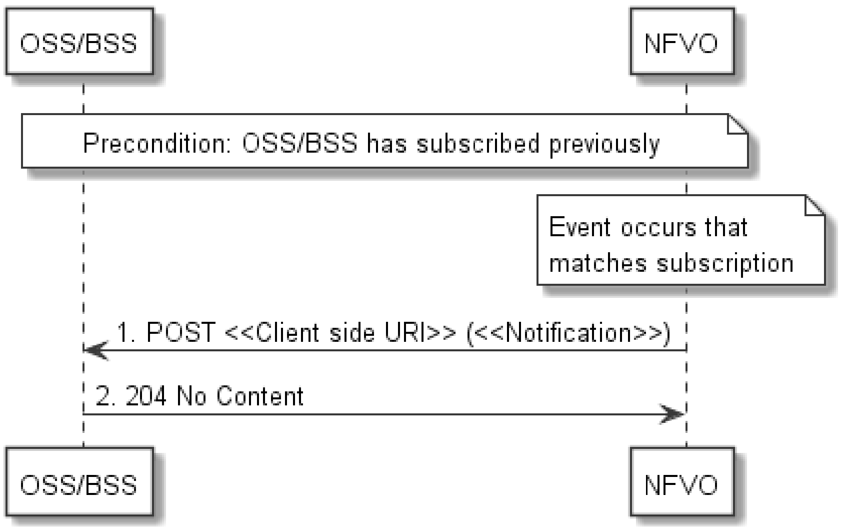
OSS/BSS
NFVO
Precondition: OSS/BSS has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URI>> (<<Notification>>)
2. 204 No Content
OSS/BSS
NFVO

Figure 8.3.4-1: Flow of sending notifications

Precondition: The OSS/BSS has subscribed previously for notifications related to NS fault management.

The procedure consists of the following steps as illustrated in Figure 8.3.4-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the NFVO 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 8.5.2.5, 8.5.2.6 and 8.5.2.7).

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

Error handling: If the NFVO does not receive the "204 No Content " response from the OSS/BSS, it can retry sending the notification.

8.4 Resources

8.4.1 Introduction

This clause defines all the resources and methods provided by the NS fault management interface.

8.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the NS fault management interface.

8.4.2 Resource: Alarms

8.4.2.1 Description

This resource represents a list of alarms related to NS instances.

8.4.2.2 Resource definition

The resource URI is:

{apiRoot}/nsfm/v1/alarms

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

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.2 GET

The client can use this method to retrieve information about the alarm list.

This method shall follow the provisions specified in the Tables 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 expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. The following attribute names shall be supported by the NFVO in the filter expression: - id - nsInstanceId - rootCauseFaultyComponent.faultyNestedNsInstanceId - rootCauseFaultyComponent.faultyNsVirtualLinkInstanceId - rootCauseFaultyComponent.faultyVnfInstanceId - rootCauseFaultyResource.faultyResourceType - eventType - perceivedSeverity - probableCause
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

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 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**
	Alarm	0..N	200 OK	Information about zero or more alarms was queried successfully. The response body shall contain the list of related alarms. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

8.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3 Resource: Individual alarm

8.4.3.1 Description

This resource represents an individual alarm.

8.4.3.2 Resource definition

The resource URI is:

{apiRoot}/nsfm/v1/alarms/{alarmId}

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.
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.	

8.4.3.3 Resource methods

8.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3.3.2 GET

The client can use this method to read an individual alarm.

This method shall follow the provisions specified in the Tables 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
none supported		

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
	Alarm	1	200 OK	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.

8.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.3.3.4 PATCH

This method modifies an individual alarm resource.

This method shall follow the provisions specified in the Tables 8.4.3.3.4-1 and 8.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.3.3.4-2: 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 8.5.2.8.
	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 8.5.2.4).
	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.

8.4.3.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4 Resource: Subscriptions

8.4.4.1 Description

This resource represents subscriptions. The client can use this resource to subscribe to notifications related to alarms related to a NS and to query its subscriptions.

8.4.4.2 Resource definition

The resource URI is:

{apiRoot}/nsfm/v1/subscriptions

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

8.4.4.3 Resource methods

8.4.4.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the Tables 8.4.4.3.1-1 and 8.4.4.3.1-2 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 OSS, and might make sense only in very rare use cases. Consequently, the NFVO 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 8.4.4.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Remarks
none supported		

Table 8.4.4.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality		Description
Request body	FmSubscriptionRequest	1	Details of the subscription to be created, as defined in clause 8.5.2.2.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	FmSubscription	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 exits and the policy of the NFVO 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.

8.4.4.3.2 GET

The client can use this method to retrieve the list of active subscriptions for alarms related to a NS 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 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	Remarks
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the FmSubscription and in data types referenced from it shall be supported by the NFVO in the filter expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

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**
	FmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e., zero or more representations of FM subscriptions, as defined in clause 8.5.2.3. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

8.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5 Resource: Individual subscription

8.4.5.1 Description

This resource represents an individual subscription for alarms related to NSs. The client can use this resource to read and to terminate a subscription to notifications related to NS fault management.

8.4.5.2 Resource definition

The resource URI is:

{apiRoot}/nsfm/v1/subscriptions/{subscriptionId}

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.
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.	

8.4.5.3 Resource methods

8.4.5.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5.3.2 GET

The client can use this method for reading an individual subscription for alarms related to NSs subscribed by the client.

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
none supported		

Table 8.4.5.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	n/a			
Response body	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.

8.4.5.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.5.3.5 DELETE

This method terminates an individual subscription.

This method shall follow the provisions specified in the Tables 8.4.5.3.5-1 and 8.4.5.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.5.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 8.4.5.3.5-2: Details of the DELETE request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	n/a			
Response body	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.

8.4.6 Resource: Notification endpoint

8.4.6.1 Description

This resource represents a notification endpoint for alarms related to NSs.

The API producer can use this resource to send notifications related to alarms related to NSs or about a rebuilt alarm list to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

8.4.6.2 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 8.4.6.2-1.

Table 8.4.6.2-1: Resource URI variables for this resource

Name	Definition
n/a	

8.4.6.3 Resource methods

8.4.6.3.1 POST

The POST method notifies an alarm related to a NS or that the alarm list has been rebuilt.

This method shall follow the provisions specified in the Tables 8.4.6.3.1-1 and 8.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 8.4.6.3.1-1: 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 8.4.6.3.1-2.

Table 8.4.6.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	AlarmNotification	1	Information of a NS alarm.	
	AlarmClearedNotification	1	Information of the clearance of a NS alarm.	
	AlarmListRebuiltNotification	1	Information that the alarm list has been rebuilt by the NFVO.	
	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.

8.4.6.3.2 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 8.4.6.3.2-1 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

	Data type	Cardinality		Description
Request body	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.

8.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

8.5 Data Model

8.5.1 Introduction

This clause defines the request and response data structures of the NS fault management interface. If a request or response contains attributes not defined in the present document, 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 Resource and notification data types

8.5.2.1 Introduction

This clause defines the data structures to be used in the resource representations and notifications for the NS fault management interface.

8.5.2.2 Type: FmSubscriptionRequest

This type represents a subscription request related to notifications about NS faults. It shall comply with the provisions defined in Table 8.5.2.2-1.

Table 8.5.2.2-1: 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 sendingnotifications 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.

8.5.2.3 Type: FmSubscription

This type represents a subscription related to notifications about NS faults. It shall comply with the provisions defined in Table 8.5.2.3-1.

Table 8.5.2.3-1: 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.

8.5.2.4 Type: Alarm

The alarm data type encapsulates information about an alarm. It shall comply with the provisions defined in Table 8.5.2.4-1.

Table 8.5.2.4-1: 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 NS instance.
rootCauseFaultyComponent	FaultyComponentInfo	1	The NS components that are causing the NS fault.
rootCauseFaultyResource	FaultyResourceInfo	0..1	The virtualised resources that are causing the NS fault. It shall be present when the faulty component is "NS Virtual Link" or "VNF" (see clause 8.5.3.4).
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.

Attribute name	Data type	Cardinality	Description
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 NS instance to which the notified alarm is correlated. Shall be present if the NS instance information is accessible as a resource.

8.5.2.5 Type: AlarmNotification

This type represents an alarm notification about NS faults. It shall comply with the provisions defined in Table 8.5.2.5-1.

Table 8.5.2.5-1: 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 NS identifier, and FaultDetails.
_links	Structure (inlined)	1	Links to resources related to this notification.
>subscription	NotificationLink	1	Link to the related subscription.

8.5.2.6 Type: AlarmClearedNotification

This type represents an alarm cleared notification about NS faults. It shall comply with the provisions defined in Table 8.5.2.6-1.

Table 8.5.2.6-1: 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	NotificationLink	1	Link to the related subscription.
>alarm	NotificationLink	1	Link to the resource that represents the related alarm.

8.5.2.7 Type: AlarmListRebuiltNotification

This type represents a notification that the alarm list has been rebuilt, e.g. if the NFVO detects its storage holding the alarm list is corrupted. It shall comply with the provisions defined in Table 8.5.2.7-1.

Table 8.5.2.7-1: 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	NotificationLink	1	Link to the related subscription.
>alarms	NotificationLink	1	Link to the alarm list, i.e. the "Alarms" resource.

8.5.2.8 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 8.5.2.4 are included in the "AlarmModifications" data type.

The "AlarmModifications" data type shall comply with the provisions defined in Table 8.5.2.8-1.

Table 8.5.2.8-1: 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

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: FmNotificationsFilter

This type represents a subscription filter related to notifications about NS faults. It shall comply with the provisions defined in Table 8.5.3.2-1.

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 8.5.3.2-1: Definition of the FmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
nsInstanceSubscriptionFilter	NsInstanceSubscription Filter	0..1	Filter criteria to select NS 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 alarms related to NSs with a faulty resource type listed in this attribute.
perceivedSeverities	PerceivedSeverityType	0..N	Match alarms related to NSs with a perceived severity listed in this attribute.
eventTypes	EventType	0..N	Match alarms related to NSs with an event type listed in this attribute.
probableCauses	String	0..N	Match alarms related to NSs 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.			

8.5.3.3 Type: FaultyResourceInfo

This type represents the faulty virtual resources that have a negative impact on a NS. It shall comply with the provisions defined in Table 8.5.3.3-1.

Table 8.5.3.3-1: 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.

8.5.3.4 Type: FaultyComponentInfo

This type represents the faulty component that has a negative impact on an NS. It shall comply with the provisions defined in Table 8.5.3.4-1.

Table 8.5.3.4-1: Definition of the FaultyComponentInfo data type

Attribute name	Data type	Cardinality	Description
faultyNestedNsInstanceId	Identifier	0..1	Identifier of the faulty nested NS instance. See note.
faultyNsVirtualLinkInstanceId	Identifier	0..1	Identifier of the faulty NS virtual link instance. See note.
faultyVnfInstanceId	Identifier	0..1	Identifier of the faulty VNF instance. See note.
NOTE: At least one of the attributes shall be present.			

8.5.4 Referenced simple data types and enumerations

8.5.4.1 Introduction

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

8.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.2.

8.5.4.3 Enumeration: PerceivedSeverityType

The enumeration PerceivedSeverityType shall comply with the provisions defined in Table 8.5.4.3-1. It indicates the relative level of urgency for operator attention.

Table 8.5.4.3-1: 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 [30]).
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 [30]).
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 [30]).
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 [30]).
INDETERMINATE	The Indeterminate severity level indicates that the severity level cannot be determined (Recommendation ITU-T X.733 [30]).
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 [30]).

8.5.4.4 Enumeration: EventType

The enumeration EventType represents those types of events that trigger an alarm. It shall comply with the provisions defined in Table 8.5.4.4-1.

Table 8.5.4.4-1: 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 [30]).
PROCESSING_ERROR_ALARM	An alarm of this type is associated with a software or processing fault (Recommendation ITU-T X.733 [30]).
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 [30]).
QOS_ALARM	An alarm of this type is associated with degradation in the quality of a service (Recommendation ITU-T X.733 [30]).
EQUIPMENT_ALARM	An alarm of this type is associated with an equipment fault (Recommendation ITU-T X.733 [30]).

8.5.4.5 Enumeration: FaultyResourceType

The enumeration FaultyResourceType represents those types of faulty resource. It shall comply with the provisions defined in Table 8.5.4.5-1.

Table 8.5.4.5-1: Enumeration FaultyResourceType

Enumeration value	Description
COMPUTE	Virtual compute resource
STORAGE	Virtual storage resource
NETWORK	Virtual network resource

9 VNF Package Management interface

9.1 Description

This interface allows the OSS/BSS to invoke VNF package management operations towards the NFVO, and to subscribe to notifications regarding VNF package on-boarding or changes provided by the NFVO.

The operations provided through this interface are as follows:

- Create VNF Package Info
- Upload VNF Package
- Update VNF Package Info
- Delete VNF Package
- QueryVNF Package Info, include obtaining the VNFD
- Fetch VNF Package
- Fetch VNF Package Artifacts
- Subscribe
- Query Subscription Info
- Notify
- Terminate Subscription

State changes of a VNF package are illustrated in clause B.2.

9.1a API version

For the VNF package management interface as specified in the present document, the MAJOR version field shall be 1, the MINOR version field shall be 1, and the PATCH version number shall be 0 (see clause 4.6.1 for a definition of the version fields). Consequently, the {apiMajorVersion} URI variable shall be set to "v1".

NOTE: The MINOR version 0 corresponds to the version of the API specified in version 2.4.1 of the present document.

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 "vnfpkgm" shall be used to represent {apiName}. 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 package management interface.

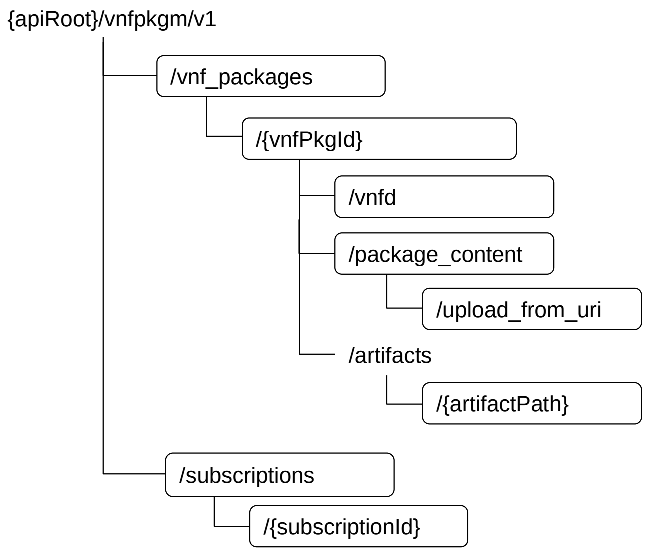

Figure 9.2-1: Resource URI structure of the VNF Package Management Interface

Table 9.2-1 lists the individual resources defined, and the applicable HTTP methods. The NFVO shall support responding to requests for all HTTP methods on the resources in Table 9.2-1 that are marked as "M" (mandatory) in the "Cat" column. The NFVO shall also support the "API versions" resources as specified in clause 4.6.3.2.

Table 9.2-1: Resources and methods overview of the VNF Package Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
VNF packages	/vnf_packages	GET	M	Query VNF packages information
		POST	M	Create a new individual VNF package resource
Individual VNF package	/vnf_packages/{vnfPkgId}	GET	M	Read information about an individual VNF package
		PATCH	M	Update information about an individual VNF package
		DELETE	M	Delete an individual VNF package
VNFD of an individual VNF package	/vnf_packages/{vnfPkgId}/vnfd	GET	M	Read VNFD of an on-boarded VNF package
VNF package content	/vnf_packages/{vnfPkgId}/package _content	GET	M	Fetch an on-boarded VNF package
		PUT	M	Upload a VNF package by providing the content of the VNF package
Upload VNF package from URI task	/vnf_packages/{vnfPkgId}/package _content/upload_from_uri	POST	M	Upload a VNF package by providing the address information of the VNF package
IndividualVNF package artifact	/vnf_packages/{vnfPkgId}/artifacts/ {artifactPath}	GET	M	Fetch individual VNF package artifact
Subscriptions	/subscriptions	POST	M	Subscribe to notifications related to on-boarding and/or changes of VNF packages
		GET	M	Query multiple subscriptions

Resource name	Resource URI	HTTP Method	Cat	Meaning
Individual subscription	/subscriptions/{subscriptionId}	GET	M	Read an individual subscription resource
		DELETE	M	Terminate a subscription
Notification endpoint	(client-provided)	POST	See note	Notify about VNF package on-boarding or change. See note
		GET	See note	Test the notification endpoint. See note
NOTE: The NFVO shall support invoking the HTTP methods defined for the "Notification endpoint" resource exposed by the OSS/BSS. If the OSS/BSS supports invoking the POST method on the "Subscriptions" resource towards the NFVO, it shall also support responding to the HTTP requests defined for the "Notification endpoint" resource.				

9.3 Sequence diagrams (informative)

9.3.1 Flow of the creation of an individual VNF package resource

This clause describes the procedure for creating an individual VNF package resource.

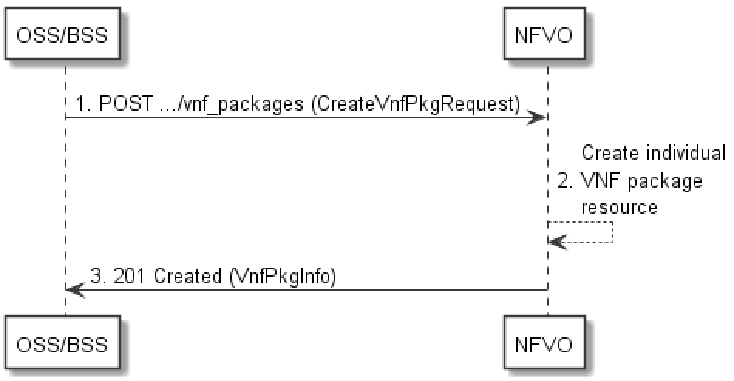
OSS/BSS
NFVO
1. POST .../vnf_packages (CreateVnfPkgRequest)
2. Create individual VNF package resource
3. 201 Created (VnfPkgInfo)
OSS/BSS
NFVO

Figure 9.3.1-1: Flow of the creation of an individual VNF package resource

Creation of an individual VNF package resource, as illustrated in Figure 9.3.1-1, consists of the following steps:

1) The OSS/BSS sends a POST request to the "VNF packages" resource including in the payload body a data structure of type "CreateVnfPkgInfoRequest".

2) The NFVO creates a new individual VNF package resource.

3) The NFVO returns a "201 Created" response containing a representation of the individual VNF package resource and a "Location" HTTP header that points to the new "individual VNF package" resource.

Postcondition: Upon successful completion, the individual VNF package resource is created with the value of the "onboardingState" attribute equals to "CREATED", the value of the "operationalState" attribute equals to "DISABLED" and the value of "usageState" attribute equals to "NOT_IN_USE".

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

9.3.2 Flow of the uploading of VNF package content

This clause describes the procedure of uploading the content of a VNF package.

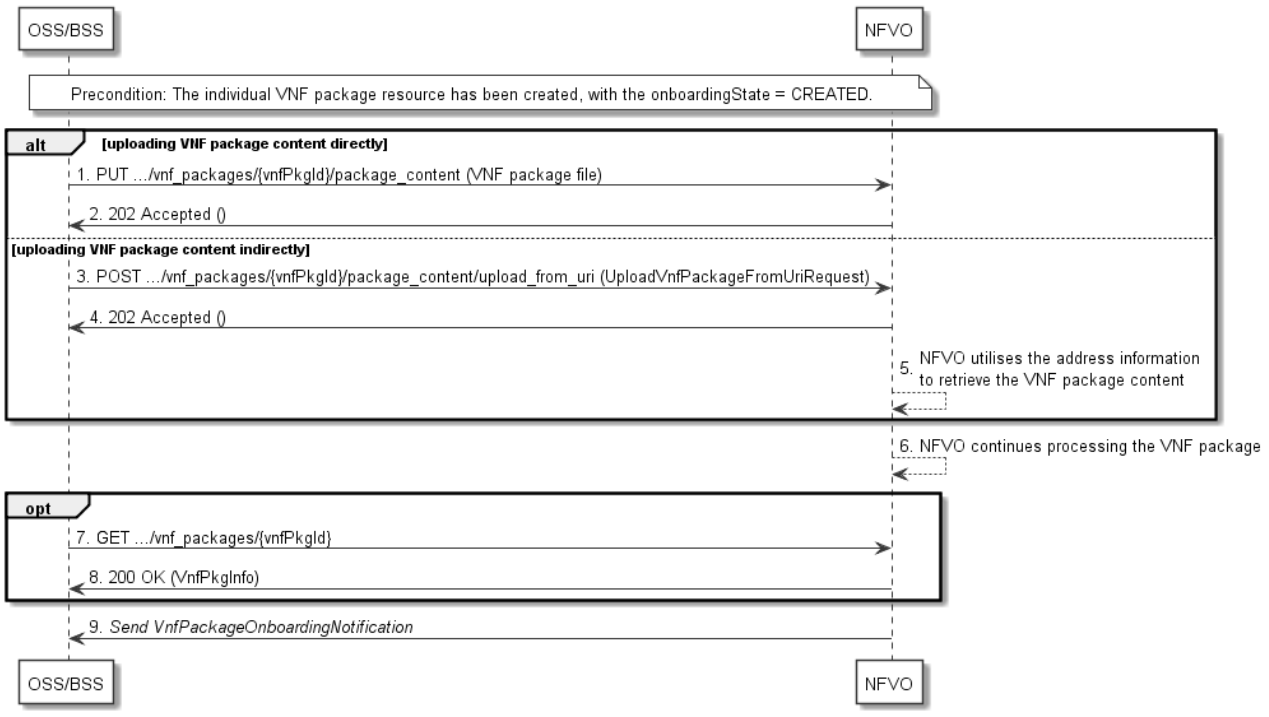
OSS/BSS
NFVO
Precondition: The individual VNF package resource has been created, with the onboardingState = CREATED.
alt
[uploading VNF package content directly]
1. PUT .../vnf_packages/{vnfPkgId}/package_content (VNF package file)
2. 202 Accepted ()
[uploading VNF package content indirectly]
3. POST .../vnf_packages/{vnfPkgId}/package_content/upload_from_uri (UploadVnfPackageFromUriRequest)
4. 202 Accepted ()
5. NFVO utilises the address information to retrieve the VNF package content
6. NFVO continues processing the VNF package
opt
7. GET .../vnf_packages/{vnfPkgId}
8. 200 OK (VnfPkgInfo)
9. Send VnfPackageOnboardingNotification
OSS/BSS
NFVO

Figure 9.3.2-1: Flow of the uploading of VNF package content

Precondition: The individual VNF package resource has been created with the value of "onboardingState" attribute equals to "CREATED".

Uploading the content of a VNF package, as illustrated in Figure 9.3.2-1, consists of the following steps:

1) If the OSS/BSS uploads the VNF package content directly to the NFVO, it sends a PUT request to the "VNF package content" resource including in the payload body a copy of the VNF package content.

2) The NFVO returns a "202 Accepted" response with an empty payload body.

3) If the OSS/BSS uploads the VNF package content indirectly to the NFVO, it sends a POST request to the "Upload VNF package from URI task" resource including in the payload body a data structure of type "UploadVnfPackageFromUriRequest".

4) The NFVO returns a "202 Accepted" response with an empty payload body to indicate the address information is successfully received.

5) The NFVO utilizes the address information to retrieve the VNF package content.

6) The NFVO continues processing the VNF package (e.g. validation) after it retrieves the package content.

7) Optionally, the OSS/BSS can send a GET request to the "individual VNF package" resource to check the on-boarding state of the VNF package resource.

8) The NFVO returns a "200 OK" response containing the information of the VNF package resource.

9) The NFVO sends a VnfPackageOnboardingNotification to the OSS/BSS to indicate the successful on-boarding of the VNF package content.

Postcondition: Upon successful completion, the content of the VNF package is on-boarded. And the state of the VNF package is changed as follows: the value of the "onboardingState" attribute equals to "ONBOARDED", the value of the "operationalState" attribute equals to "ENABLED" and the value of the "usageState" attribute equals to "NOT_IN_USE".

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

9.3.3 Flow of querying/reading VNF package information

This clause describes a sequence for querying information about one or multiple VNF packages.

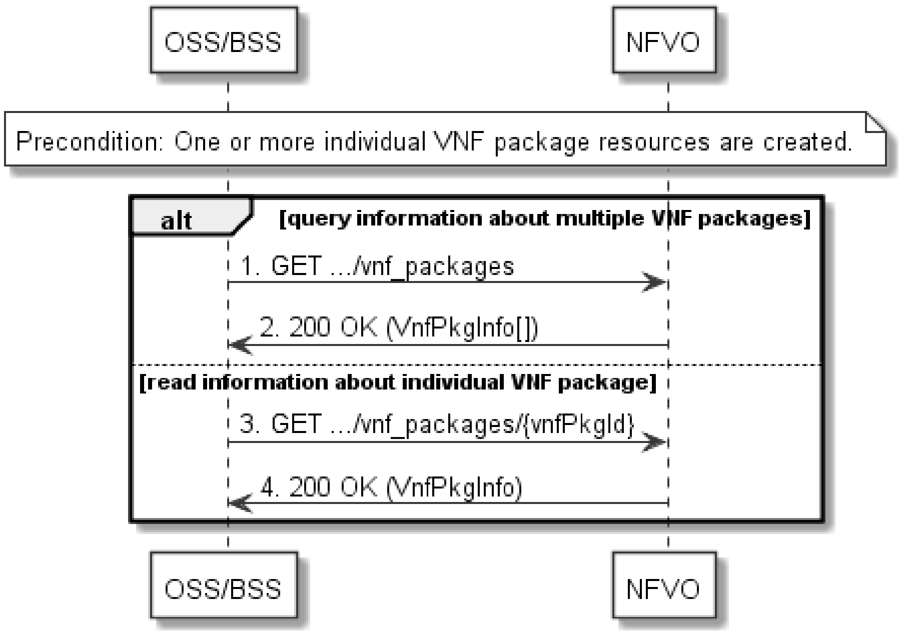
OSS/BSS
NFVO
Precondition: One or more individual VNF package resources are created.
alt
[query information about multiple VNF packages]
1. GET .../vnf_packages
2. 200 OK (VnfPkgInfo[])
[read information about individual VNF package]
3. GET .../vnf_packages/{vnfPkgId}
4. 200 OK (VnfPkgInfo)
OSS/BSS
NFVO

Figure 9.3.3-1: Flow of querying/reading VNF package information

Precondition: One or more individual VNF package resources are created.

VNF package information query, as illustrated in Figure 9.3.3-1, consists of the following steps:

1) If the OSS/BSS intends to query information about multiple VNF packages, it sends a GET request to the "VNF packages" resource.

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

3) If the OSS/BSS intends to read information about a particular VNF package, the OSS/BSS sends a GET request to the "Individual VNF package" resource, addressed by the appropriate VNF package identifier in its resource URI.

4) The NFVO returns a "200 OK" response, and includes in the payload body a data structure of type "VnfPkgInfo".

Postcondition: Upon successful completion, the OSS/BSS gets the information of the VNF packages or the VNF package.

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

9.3.4 Flow of reading the VNFD of an on-boarded VNF package

This clause describes the procedure for reading the VNFD of an on-boarded VNF package.

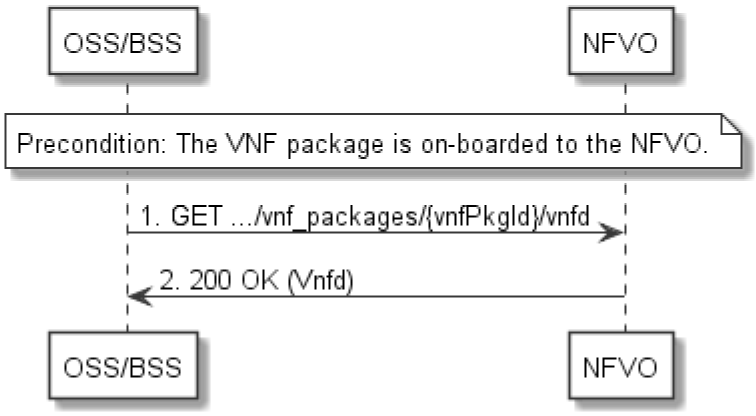
OSS/BSS
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
1. GET .../vnf_packages/{vnfPkgId}/vnfd
2. 200 OK (Vnfd)
OSS/BSS
NFVO

Figure 9.3.4-1: Flow of reading VNFD

Precondition: The VNF package is on-boarded to the NFVO.

The procedure consists of the following steps as illustrated in Figure 9.3.4-1.

1) The OSS/BSS sends a GET request to the "VNFD in an individual VNF package" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the VNFD from the VNF package in the payload body.

9.3.5 Flow of updating information of a VNF package

This clause describes the procedure for enabling/disabling/abort deletion of a VNF package.

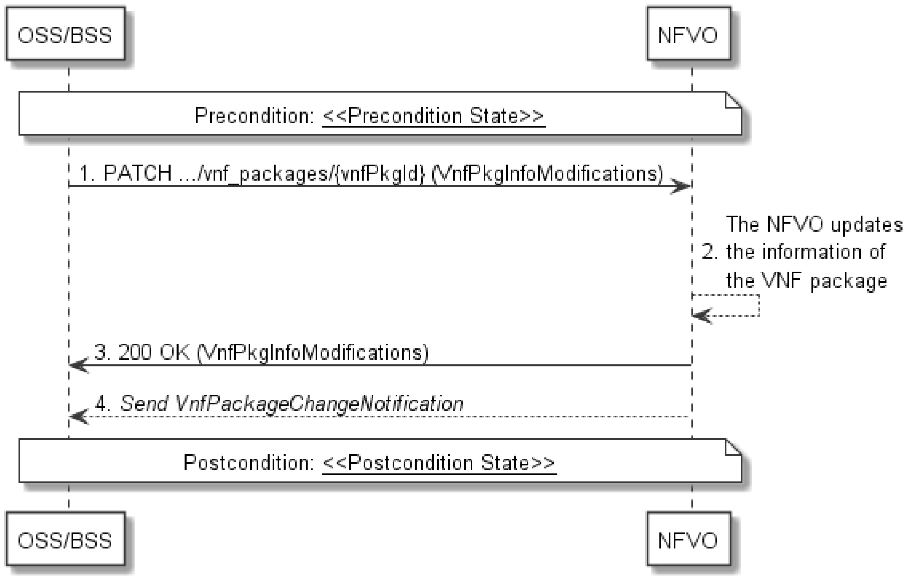
OSS/BSS
NFVO
Precondition: <<Precondition State>>
1. PATCH .../vnf_packages/{vnfPkgId} (VnfPkgInfoModifications)
2. The NFVO updates the information of the VNF package
3. 200 OK (VnfPkgInfoModifications)
4. Send VnfPackageChangeNotification
Postcondition: <<Postcondition State>>
OSS/BSS
NFVO

Figure 9.3.5-1: Flow of updating information of a VNF package

NOTE: Due to possible race conditions, the 200 response and the VnfPackageChangeNotification can arrive in any order at the OSS/BSS.

Precondition: The VNF package is in <<Precondition State>>, the value of <<Precondition State>> depends on the actual requested operation, and is described in Table 9.3.5-1.

The procedure consists of the following steps as illustrated in Figure 9.3.5-1:

1) The OSS/BSS sends a PATCH request to the "individual VNF package" resource including in the payload body a data structure of type "VnfPkgInfoModifications".

2) The NFVO updates the information of the VNF package.

3) The NFVO returns a "200 OK" response with a payload body containing a data structure of type "VnfPkgInfoModifications".

4) If the operational state of the VNF package is modified, the NFVO sends to OSS/BSS a VnfPackageChangeNotification to indicate the state change of the VNF package.

Postcondition: The VNF package is in << Postcondition State>>.

Table 9.3.5-1 describes how the <<Precondition State>> and << Postcondition State>> are parameterized in the above flow.

Table 9.3.5-1: Parameterization of the flow for updating information of a VNF package

Operation	<<Precondition State>>	<<PostconditionState>>
Enable a VNF package	The on-boarding state of the VNF package is ONBOARDED and the operational state of the VNF package is DISABLED	The operational state of the VNF package is ENABLED
Disable a VNF package	The on-boarding state of the VNF package is ONBOARDED and the operational state of the VNF package is ENABLED	The operational state of the VNF package is DISABLED
Update user defined data	The individual VNF package resource is created	The user defined data is updated

9.3.6 Flow of deleting a VNF package resource

This clause describes a sequence for deleting a VNF package resource.

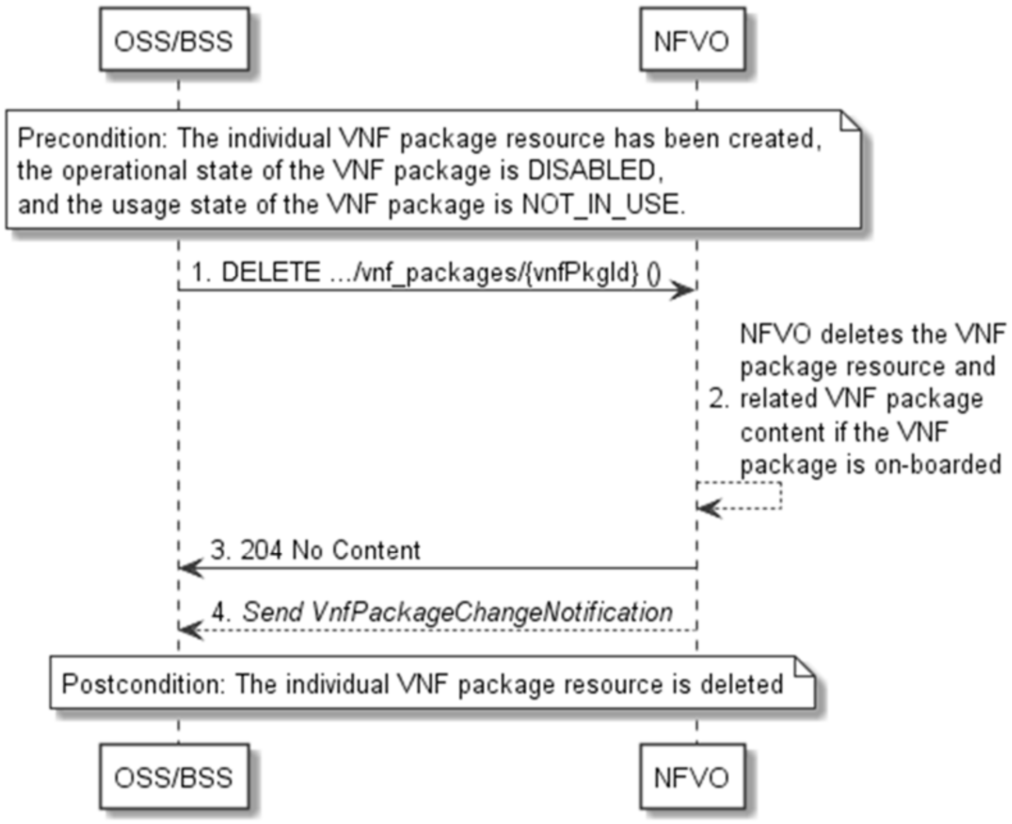
OSS/BSS
NFVO
Precondition: The individual VNF package resource has been created, the operational state of the VNF package is DISABLED, and the usage state of the VNF package is NOT_IN_USE.
1. DELETE .../vnf_packages/{vnfPkgId} ()
2. NFVO deletes the VNF package resource and related VNF package content if the VNF package is on-boarded
3. 204 No Content
4. Send VnfPackageChangeNotification
Postcondition: The individual VNF package resource is deleted
OSS/BSS
NFVO

Figure 9.3.6-1: Flow of deleting a VNF package resource

NOTE: Due to possible race conditions, the 204 response and the VnfPackageChangeNotification can arrive in any order at the OSS/BSS.

Precondition: The individual VNF package resource has been created, the operational state of the VNF package is DISABLED, and the usage state of the VNF package is NOT_IN_USE.

Deleting a VNF package resource, as illustrated in Figure 9.3.6-1, consists of the following steps:

1) The OSS/BSS sends a DELETE request to the "individual VNF package" resource.

2) The NFVO deletes the "individual VNF package" resource and related VNF package content if the VNF package is on-boarded.

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

4) The NFVO sends to the OSS/BSS a VnfPackageChangeNotification to indicate the deletion of the VNF package resource.

Postcondition: Upon successful completion, the individual VNF package resource is deleted.

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

9.3.7 Flow of fetching an on-boarded VNF package

This clause describes a sequence for fetching the content of an on-boarded VNF package.

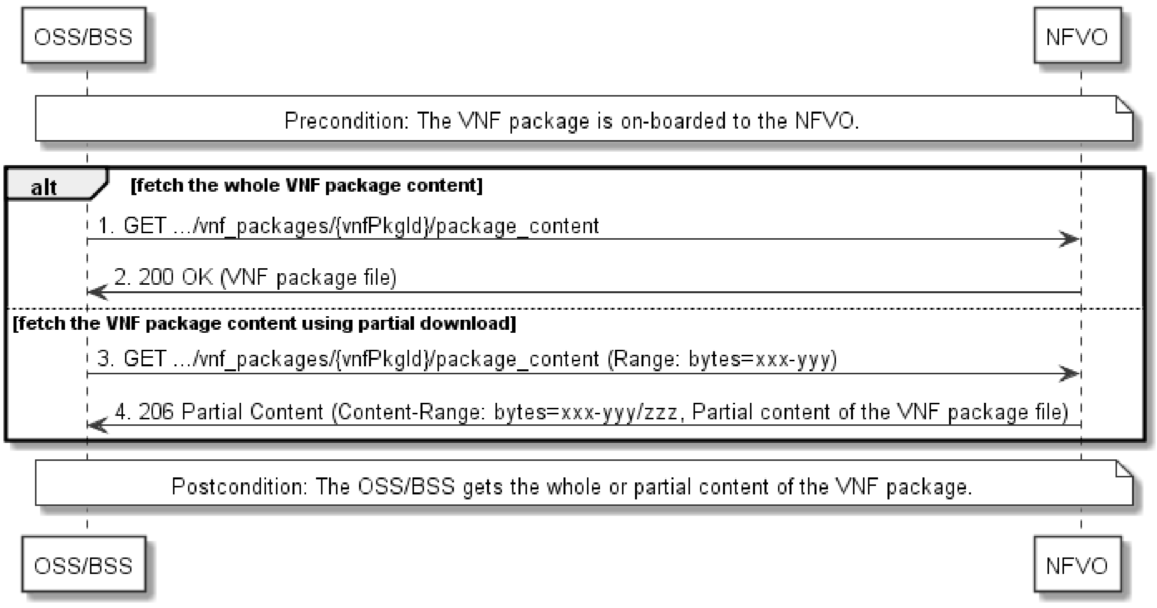
OSS/BSS
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
alt
[fetch the whole VNF package content]
1. GET .../vnf_packages/{vnfPkgId}/package_content
2. 200 OK (VNF package file)
[fetch the VNF package content using partial download]
3. GET .../vnf_packages/{vnfPkgId}/package_content (Range: bytes=xxx-yyy)
4. 206 Partial Content (Content-Range: bytes=xxx-yyy/zzz, Partial content of the VNF package file)
Postcondition: The OSS/BSS gets the whole or partial content of the VNF package.
OSS/BSS
NFVO

Figure 9.3.7-1: Flow of fetching an on-boarded VNF package

Precondition: The VNF package is on-boarded to the NFVO.

Fetching an on-boarded VNF package, as illustrated in Figure 9.3.7-1, consists of the following steps.

1) If fetching the whole VNF package content, the OSS/BSS sends a GET request to the "VNF package content" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the VNF package file in the payload body.

3) If fetching the VNF package content using partial download, the OSS/BSS sends a GET request to the "VNF package content" resource, and includes a "Range" HTTP header indicating the partition of the VNF package content needs to be transferred.

4) The NFVO returns a "206 Partial Content" response with a payload body containing the partial content of the VNF package, and a "Content-Range" HTTP header indicating the byte range enclosed in the payload and the complete length of the VNF package content.

Postcondition: Upon successful completion, the OSS/BSS gets the whole or partial content of the VNF package.

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

9.3.8 Flow of fetching a VNF package artifact

This clause describes a sequence for fetching an individual artifact contained in an on-boarded VNF package.

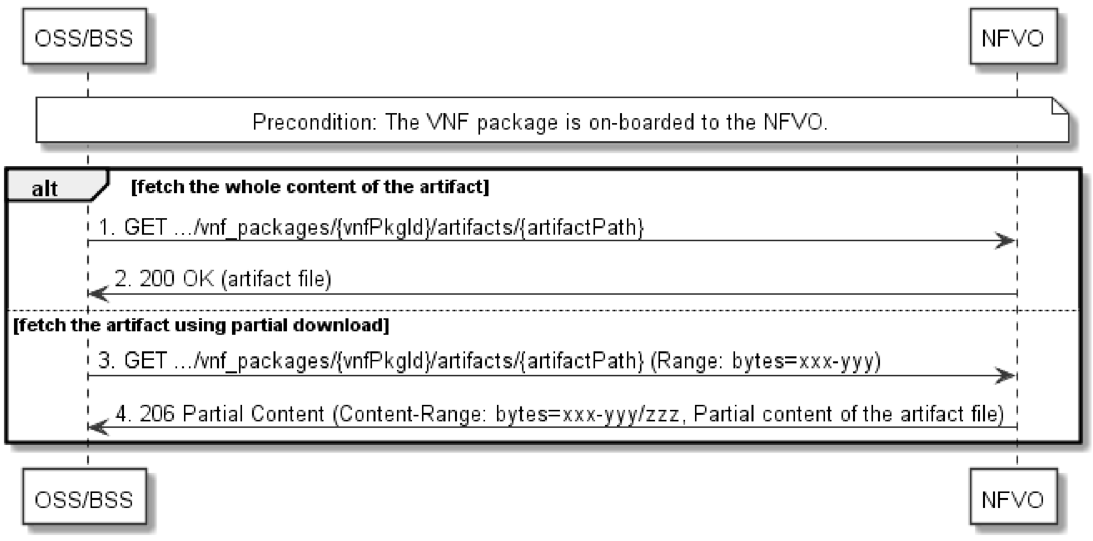
OSS/BSS
NFVO
Precondition: The VNF package is on-boarded to the NFVO.
alt
[fetch the whole content of the artifact]
1. GET .../vnf_packages/{vnfPkgId}/artifacts/{artifactPath}
2. 200 OK (artifact file)
[fetch the artifact using partial download]
3. GET .../vnf_packages/{vnfPkgId}/artifacts/{artifactPath} (Range: bytes=xxx-yyy)
4. 206 Partial Content (Content-Range: bytes=xxx-yyy/zzz, Partial content of the artifact file)
OSS/BSS
NFVO

Figure 9.3.8-1: Flow of fetching a VNF package artifact

Precondition: The VNF package is on-boarded to the NFVO.

Fetching an individual artifact contained in an on-boarded VNF package, as illustrated in Figure 9.3.8-1, consists of the following steps.

1) If fetching the whole content of the artifact, the OSS/BSS sends a GET request to the "Individual VNF package artifact" resource.

2) The NFVO returns a "200 OK" response, and includes a copy of the applicable artifact file from the VNF package in the payload body.

3) If fetching the artifact using partial download, the OSS/BSS sends a GET request to the "Individua VNF package artifact" resource, and includes a "Range" HTTP header indicating the partition of the artifact needs to be transferred.

4) The NFVO returns a "206 Partial Content" response with a payload body containing the partial content of the artifact file, and a "Content-Range" HTTP header indicating the byte range enclosed in the payload and the complete length of the artifact file.

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

9.3.9 Flow of managing subscriptions

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to VNF package management.

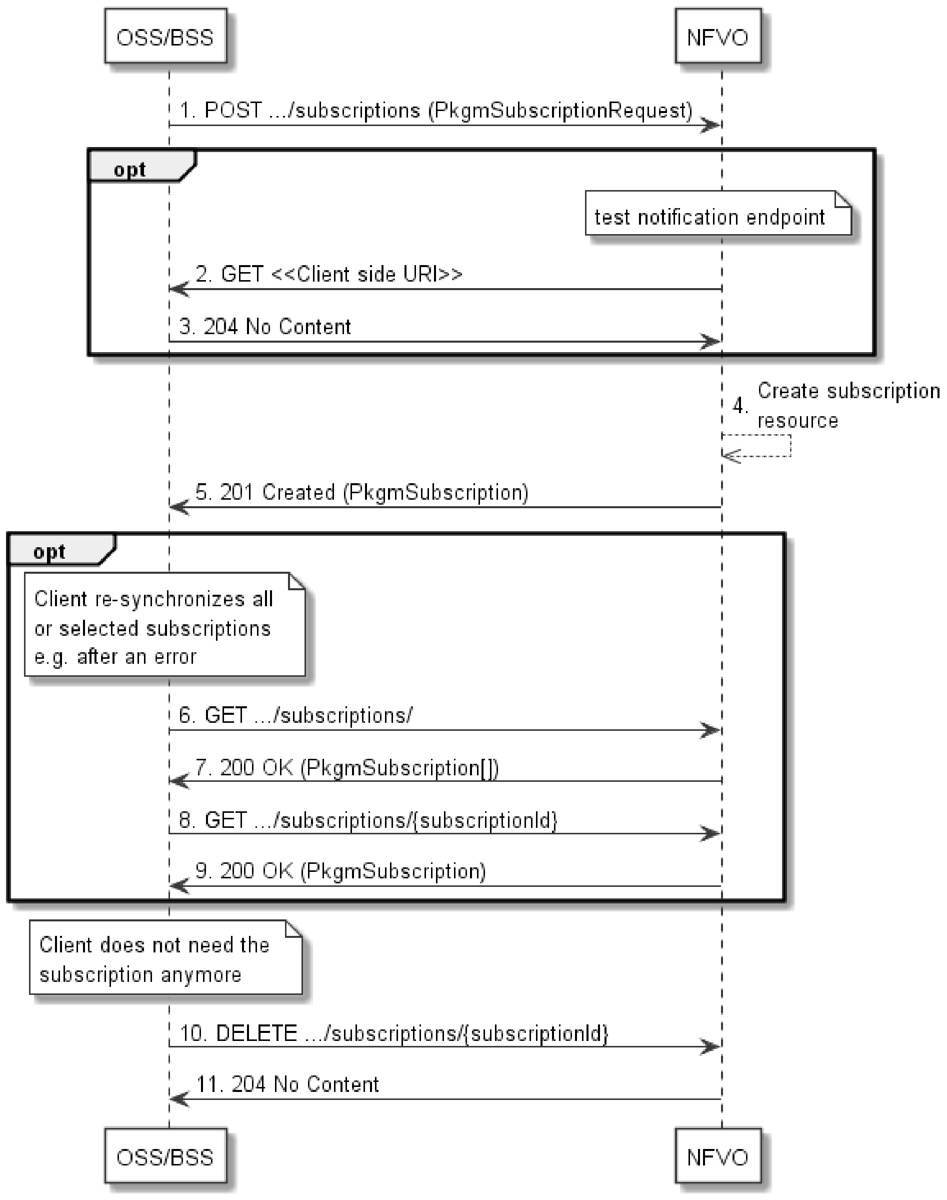
OSS/BSS
NFVO
1. POST .../subscriptions (PkgmSubscriptionRequest)
opt
test notification endpoint
2. GET <<Client side URI>>
3. 204 No Content
4. Create subscription resource
5. 201 Created (PkgmSubscription)
opt
Client re-synchronizes all or selected subscriptions e.g. after an error
6. GET .../subscriptions/
7. 200 OK (PkgmSubscription[])
8. GET .../subscriptions/{subscriptionId}
9. 200 OK (PkgmSubscription)
Client does not need the subscription anymore
10. DELETE .../subscriptions/{subscriptionId}
11. 204 No Content
OSS/BSS
NFVO

Figure 9.3.9-1: Flow of managing subscriptions

The procedure consists of the following steps as illustrated in Figure 9.3.9-1:

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

2) Optionally, to test the notification endpoint that was registered by the OSS/BSS as part of the subscription, the NFVO sends a GET request to the notification endpoint URI.

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

4) The NFVO creates a new subscription to notifications related to VNF package on-boarding or changes, and a resource that represents this subscription.

5) The NFVO returns a "201 Created" response containing a data structure of type "PkgmSubscription" representing the subscription resource just created by the NFVO, 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 OSS/BSS may obtain information about its subscriptions by sending a GET request to the resource representing the subscriptions.

7) In that case, the NFVO returns a "200 OK" response that contains the list of representations of all existing subscriptions that were created by the OSS/BSS.

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

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

10) If the OSS/BSS 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 NFVO acknowledges the successful termination of the subscription by returning a "204 No Content" response.

Error handling: The NFVO rejects a subscription if the subscription information is not valid: endpoint cannot be reached, subscription information is malformed, etc.

9.3.10 Flow of sending notifications

This clause describes the procedure for sending notifications related to VNF package management.

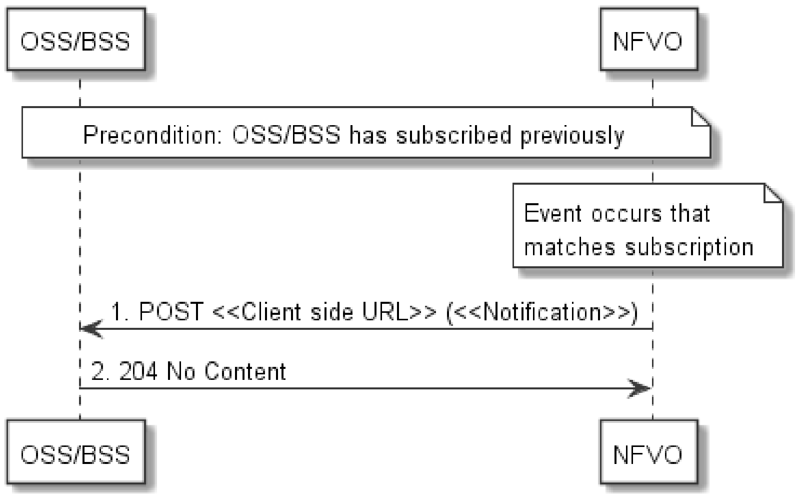
OSS/BSS
NFVO
Precondition: OSS/BSS has subscribed previously
Event occurs that matches subscription
1. POST <<Client side URL>> (<<Notification>>)
2. 204 No Content
OSS/BSS
NFVO

Figure 9.3.10-1: Flow of sending notifications

Precondition: The OSS/BSS has subscribed previously for notifications related to VNF package management.

The procedure consists of the following steps as illustrated in Figure 9.3.10-1:

1) If an event occurs that matches the filtering criteria which are part of the subscription, the NFVO generates a notification that includes information about the event, and sends it in the body of a POST request to the URI which the OSS/BSS 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 9.5.2.7 and 9.5.2.8).

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

Error handling: If the NFVO does not receive the "204 No Content" response from the OSS/BSS, it can retry sending the notification.

9.4 Resources

9.4.1 Introduction

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

9.4.1a Resource: API versions

The "API versions" resources as defined in clause 4.6.3.3 are part of the VNF package management interface.

9.4.2 Resource: VNF packages

9.4.2.1 Description

This resource represents VNF packages. The client can use this resource to create individual VNF package resources, and to query information of the VNF packages.

9.4.2.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages

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

The POST method creates a new individual VNF package resource.

This method shall follow the provisions specified in the Tables 9.4.2.3.1-1 and 9.4.2.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.2.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.2.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	CreateVnfPkgInfoRequest	1	IndividualVNF package resource creation parameters, as defined in clause 9.5.2.2.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfPkgInfo	1	201 Created	An individual VNF package resource has been created successfully. The response body shall contain a representation of the new individual VNF package resource, as defined in clause 9.5.2.4. The HTTP response shall include a "Location" HTTP header that contains the resource URI of the individual VNF package 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.

9.4.2.3.2 GET

The GET method queries the information of the VNF packages matching the filter.

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
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this parameter as part of the URI query string. The OSS/BSS may supply this parameter. All attribute names that appear in the VnfPkgInfo and in data types referenced from it shall be supported by the NFVO in the filter expression.
all_fields	0..1	Include all complex attributes in the response. See clause 4.3.3 for details. The NFVO shall support this parameter.
fields	0..1	Complex attributes to be included into the response. See clause 4.3.3 for details. The NFVO should support this parameter.
exclude_fields	0..1	Complex attributes to be excluded from the response. See clause 4.3.3 for details. The NFVO 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 NFVO shall support this parameter. The following attributes shall be excluded from the VnfPkgInfo structure in the response body if this parameter is provided, or none of the parameters "all_fields," "fields", "exclude_fields", "exclude_default" are provided: - softwareImages - additionalArtifacts - userDefinedData - checksum
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 9.4.2.3.2-2: 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**
	VnfPkgInfo	0..N	200 OK	Information about zero or more VNF packages was successfully queried. The response body shall contain in an array the VNF package info representations that match the attribute filter, i.e., zero or more VNF package info representations as defined in clause 9.5.2.5. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. In the returned ProblemDetails structure, 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	1	400 Bad Request	Error: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

9.4.2.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.2.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.2.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3 Resource: Individual VNF package

9.4.3.1 Description

This resource represents an individual VNF package. The client can use this resource to read information of the VNF package, update information of the VNF package, or delete a VNF package.

9.4.3.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}

This resource shall support the resource URI variables defined in Table 9.4.3.2-1.

Table 9.4.3.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "VnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

9.4.3.3 Resource methods

9.4.3.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3.3.2 GET

The GET method reads the information of an individual VNF package.

This method shall follow the provisions specified in the Tables 9.4.3.3.2-1 and 9.4.3.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.3.3.2-1: 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
	vnfPkgInfo	1	200 OK	Information of the VNF package was read successfully. The response body shall contain the VNF package info representation defined in clause 9.5.2.5.
	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.

9.4.3.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.3.3.4 PATCH

The PATCH method updates the information of a VNF package.

This method shall follow the provisions specified in the Tables 9.4.3.3.4-1 and 9.4.3.3.4-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.3.3.4-1: URI query parameters supported by the PATCH method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.3.3.4-2: Details of the PATCH request/response on this resource

	Data type	Cardinality		Description
Request body	VnfPkgInfoModifications	1		Parameters for VNF package information modifications.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	VnfPkgInfoModifications	1	200 OK	The operation was completed successfully. The response body shall contain attribute modifications for an "Individual VNF package" resource.
	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 any of the following scenarios: - Disable a VNF package resource of which the operational state is not ENABLED - Enable a VNF package resource of which the operational state is not DISABLED 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.

9.4.3.3.5 DELETE

The DELETE method deletes an individual VNF package resource.

This method shall follow the provisions specified in the Tables 9.4.3.3.5-1 and 9.4.3.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.3.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.3.3.5-2: 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 package was 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 operational state of the VNF package resource is ENABLED or there are running VNF instances which are instantiated based on the concerned VNF package. 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.

9.4.4 Resource: VNFD in an individual VNF package

9.4.4.1 Description

This resource represents the VNFD contained in an on-boarded VNF package. The client can use this resource to obtain the content of the VNFD.

9.4.4.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/vnfd

This resource shall support the resource URI variables defined in Table 9.4.4.2-1.

Table 9.4.4.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

9.4.4.3 Resource methods

9.4.4.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.4.3.2 GET

The GET method reads the content of the VNFD within a VNF package.

The VNFD can be implemented as a single file or as a collection of multiple files. If the VNFD is implemented in the form of multiple files, a ZIP file embedding these files shall be returned. If the VNFD is implemented as a single file, either that file or a ZIP file embedding that file shall be returned.

The selection of the format is controlled by the "Accept" HTTP header passed in the GET request.

- If the "Accept" header contains only "text/plain" and the VNFD is implemented as a single file, the file shall be returned; otherwise, an error message shall be returned.
- If the "Accept" header contains only "application/zip", the single file or the multiple files that make up the VNFD shall be returned embedded in a ZIP file.
- If the "Accept" header contains both "text/plain" and "application/zip", it is up to the NFVO to choose the format to return for a single-file VNFD; for a multi-file VNFD, a ZIP file shall be returned.

The default format of the ZIP file shall be the one specified in ETSI GS NFV-SOL 004 [5] where only the YAML files representing the VNFD, and information necessary to navigate the ZIP file and to identify the file that is the entry point for parsing the VNFD (such as TOSCA-meta or manifest files or naming conventions) are included.

This method shall follow the provisions specified in the Tables 9.4.4.3.2-1 and 9.4.4.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.4.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.4.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a		The request shall contain the appropriate entries in the "Accept" HTTP header as defined above.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, the content of the VNFD is returned. The payload body shall contain a copy of the file representing the VNFD or a ZIP file that contains the file or multiple files representing the VNFD, as specified above. The "Content-Type" HTTP header shall be set according to the format of the returned file, i.e. to "text/plain" for a YAML file or to "application/zip" for a ZIP file.
	ProblemDetails	0..1	406 Not AccepTabl e	If the "Accept" header does not contain at least one name of a content type for which the NFVO can provide a representation of the VNFD, the NFVO shall respond with this response code. The "ProblemDetails" structure may be included with the "detail" attribute providing 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 "onboardingState" of the VNF package has a value different from "ONBOARDED". 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.

9.4.4.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.4.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.4.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.5 Resource: VNF package content

9.4.5.1 Description

This resource represents a VNF package identified by the VNF package identifier allocated by the NFVO. The client can use this resource to fetch the content of the VNF package.

9.4.5.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content

This resource shall support the resource URI variables defined in Table 9.4.5.2-1.

Table 9.4.5.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note.
NOTE: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification.	

9.4.5.3 Resource methods

9.4.5.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.5.3.2 GET

The GET method fetches the content of a VNF package identified by the VNF package identifier allocated by the NFVO.

This method shall follow the provisions specified in the Tables 9.4.5.3.2-1 and 9.4.5.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.5.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.5.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	n/a		The request may contain a "Range" HTTP header to obtain single range of bytes from the VNF package file. This can be used to continue an aborted transmission. If the NFVO does not support range requests, it should return the whole file with a 200 OK response instead.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, a copy of the VNF package file is returned. The response body shall include a copy of the VNF package file. The "Content-Type" HTTP header shall be set according to the type of the file, i.e. to "application/zip" for a VNF Package as defined in ETSI GS NFV-SOL 004 [5].
	n/a	1	206 Partial Content	On success, if the NFVO supports range requests, a single consecutive byte range from the content of the VNF package file is returned. The response body shall contain the requested part of the VNF package file. The "Content-Range" HTTP header shall be provided according to IETF RFC 7233 [23]. The "Content-Type" HTTP header shall be set as defined above for the "200 OK" response.
	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 "onboardingState" of the VNF package has a value different from "ONBOARDED". The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	416 Range Not Satisfiable	The byte range passed in the "Range" header did not match any available byte range in the VNF package file (e.g. "access after end of file"). The response body may contain a ProblemDetails structure.
	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.

9.4.5.3.3 PUT

The PUT method uploads the content of a VNF package.

This method shall follow the provisions specified in the Tables 9.4.5.3.3-1 and 9.4.5.3.3-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.5.3.3-1: URI query parameters supported by the PUT method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.5.3.3-2: Details of the PUT request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	**Data type**	**Cardinality**		**Description**
	n/a	1		The payload body contains a ZIP file that represents the VNF package. The "Content-Type" HTTP header shall be set according to the type of the file, i.e. to "application/zip" for a VNF Package as defined in ETSI GS NFV-SOL 004 [5].
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The VNF package was accepted for uploading, but the processing has not been completed. It is expected to take some time for processing. The response body shall be empty. See note.
	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 onboarding state of the VNF package resource is not CREATED . 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.
NOTE: The client can track the uploading progress by receiving the "VnfPackageOnBoardingNotification" from the NFVO or by reading the status of the individual VNF package resource using the GET method.				

9.4.5.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.5.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.6 Resource: Upload VNF package from URI task

9.4.6.1 Description

This task resource represents the "Upload VNF package from URI" operation. The client can use this resource to request the uploading of a VNF package by providing address information to the NFVO for retrieving the content of the VNF package.

9.4.6.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content/upload_from_uri

This resource shall support the resource URI variables defined in Table 9.4.6.2-1.

Table 9.4.6.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. 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 Individual VNF package resource.	

9.4.6.3 Resource methods

9.4.6.3.1 POST

The POST method provides the information for the NFVO to get the content of a VNF package.

This method shall follow the provisions specified in the Tables 9.4.6.3.1-1 and 9.4.6.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.6.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.6.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Response Codes	Description
Request body	UploadVnfPkgFromUriRequest	1		The payload body contains the address information based on which the NFVO can obtain the content of the VNF package.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a		202 Accepted	The information about the VNF package was received successfully, but the on-boarding has not been completed. It is expected to take some time for processing. The response body shall be empty. See note.
	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 on-boarding state of the VNF package resource is not CREATED. 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.
NOTE: The client can track the uploading progress by receiving the "VnfPackageOnBoardingNotification" from the NFVO or by reading the status of the individual on-boarded VNF package resource using the GET method.				

9.4.6.3.2 GET

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.6.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.6.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.6.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.7 Resource: Individual VNF package artifact

9.4.7.1 Description

This resource represents an individual artifact contained in a VNF package. The client can use this resource to fetch the content of the artifact.

9.4.7.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/vnf_packages/{vnfPkgId}/artifacts/{artifactPath}

This resource shall support the resource URI variables defined in Table 9.4.7.2-1.

Table 9.4.7.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.
vnfPkgId	Identifier of the VNF package. The identifier is allocated by the NFVO. See note 1.
artifactPath	Sequence of one or path segments representing the path of the artifact within the VNF package. See note 2. EXAMPLE: foo/bar/run.sh
NOTE 1: This identifier can be retrieved from the "vnfPkgId" attribute in the VnfPackageOnboardingNotification or VnfPackageChangeNotification. NOTE 2: This identifier can be retrieved from the "artifactPath" attribute of the applicable "additionalArtifacts" entry in the body of the response to a GET request querying the "Individual VNF package" or the "VNF packages" resource.	

9.4.7.3 Resource methods

9.4.7.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.7.3.2 GET

The GET method fetches the content of an artifact within a VNF package.

This method shall follow the provisions specified in the Tables 9.4.7.3.2-1 and 9.4.7.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.7.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.7.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	n/a			The request may contain a "Range" HTTP header to obtain single range of bytes from an artifact file. This can be used to continue an aborted transmission. If the NFVO does not support range requests, it should return the whole file with a 200 OK response instead.
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	n/a	1	200 OK	On success, the content of the artifact is returned. The payload body shall contain a copy of the artifact file from the VNF package, as defined by ETSI GS NFV-SOL 004 [5]. The "Content-Type" HTTP header shall be set according to the content type of the artifact file. If the content type cannot be determined, the header shall be set to the value "application/octet-stream".
	n/a	1	206 Partial Content	A single consecutive byte range from the content of the artifact file, if the NFVO supports range requests. The response body shall contain the requested part of the artifact file from the VNF package, as defined by ETSI GS NFV-SOL 004 [5]. The "Content-Type" HTTP header shall be set according to the content type of the artifact file. If the content type cannot be determined, the header shall be set to the value "application/octet-stream". The "Content-Range" HTTP header shall be provided according to IETF RFC 7233 [23].
	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 "onboardingState" of the VNF package has a value different from "ONBOARDED". The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	416 Range Not Satisfiable	The byte range passed in the "Range" header did not match any available byte range in the artifact file (e.g. "access after end of file"). The response body may contain a ProblemDetails structure.
	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.

9.4.7.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.7.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.7.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.8 Resource: Subscriptions

9.4.8.1 Description

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

9.4.8.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/subscriptions

This resource shall support the resource URI variables defined in Table 9.4.8.2-1.

Table 9.4.8.2-1: Resource URI variables for this resource

Name	Definition
apiRoot	See clause 4.2.

9.4.8.3 Resource methods

9.4.8.3.1 POST

The POST method creates a new subscription.

This method shall follow the provisions specified in the Tables 9.4.8.3.1-1 and 9.4.8.3.1-2 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 OSS, and might make sense only in very rare use cases. Consequently, the NFVO 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 9.4.8.3.1-1: URI query parameters supported by the POST method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.8.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	PkgmSubscriptionRequest	1	Details of the subscription to be created.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PkgmSubscription	1	201 Created	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 exits and the policy of the NFVO 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.

9.4.8.3.2 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 9.4.8.3.2-1 and 9.4.8.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.8.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
filter	0..1	Attribute-based filtering expression according to clause 4.3.2. The NFVO shall support receiving this filtering parameter as part of the URI query string. The OSS/BSS may supply this filtering parameter. All attribute names that appear in the PkgmSubscription and in data types referenced from it shall be supported by the NFVO in the filtering expression.
nextpage_opaque_marker	0..1	Marker to obtain the next page of a paged response. Shall be supported by the NFVO if the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource.

Table 9.4.8.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality		Description
Request body	**Data type**	**Cardinality**	**Description**	
	n/a			
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	PkgmSubscription	0..N	200 OK	The list of subscriptions was queried successfully. The response body shall contain in an array the representations of all active subscriptions of the functional block that invokes the method, i.e., zero or more representations of VNF package management subscriptions, as defined in clause 9.5.2.7. If the NFVO supports alternative 2 (paging) according to clause 4.7.2.1 for this resource, inclusion of the Link HTTP header in this response shall follow the provisions in clause 4.7.2.3.
	ProblemDetails	1	400 Bad Request	Error: Invalid attribute-based filtering expression. 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: Response too big. If the NFVO supports alternative 1 (error) according to clause 4.7.2.1 for this resource, this error response shall follow the provisions in clause 4.7.2.2.
	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.

9.4.8.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.8.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.8.3.5 DELETE

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.9 Resource: Individual subscription

9.4.9.1 Description

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

9.4.9.2 Resource definition

The resource URI is:

{apiRoot}/vnfpkgm/v1/subscriptions/{subscriptionId}

This resource shall support the resource URI variables defined in Table 9.4.9.2-1.

Table 9.4.9.2-1: 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.	

9.4.9.3 Resource methods

9.4.9.3.1 POST

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.9.3.2 GET

The GET method reads an individual subscription.

This method shall follow the provisions specified in the Tables 9.4.9.3.2-1 and 9.4.9.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.9.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.9.3.2-2: 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
	PkgmSubscription	1	200 OK	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.

9.4.9.3.3 PUT

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.9.3.4 PATCH

This method is not supported. When this method is requested on this resource, the NFVO shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.9.3.5 DELETE

The DELETE method terminates an individual subscription.

This method shall follow the provisions specified in the Tables 9.4.9.3.5-1 and 9.4.9.3.5-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.9.3.5-1: URI query parameters supported by the DELETE method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.9.3.5-2: 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.
	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.

9.4.10 Resource: Notification endpoint

9.4.10.1 Description

This resource represents a notification endpoint.

The API producer can use this resource to send notifications related to VNF package management events to a subscribed API consumer, which has provided the URI of this resource during the subscription process.

9.4.10.2 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 9.4.10.2-1.

Table 9.4.10.2-1: Resource URI variables for this resource

Name	Definition
n/a	

9.4.10.3 Resource methods

9.4.10.3.1 POST

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

This method shall follow the provisions specified in the Tables 9.4.10.3.1-1 and 9.4.10.3.1-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.10.3.1-1: 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 9.4.10.3.1-2.

Table 9.4.10.3.1-2: Details of the POST request/response on this resource

	Data type	Cardinality	Description	
Request body	VnfPackageOnboardingNotification	1	A notification about on-boarding of a VNF package.	
	VnfPackageChangeNotification	1	A notification about changes of status in a VNF package.	
Response body	**Data type**	**Cardinality**	**Response Codes**	**Description**
	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.

9.4.10.3.2 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 9.4.10.3.2-1 and 9.4.10.3.2-2 for URI query parameters, request and response data structures, and response codes.

Table 9.4.10.3.2-1: URI query parameters supported by the GET method on this resource

Name	Cardinality	Description
none supported		

Table 9.4.10.3.2-2: Details of the GET request/response on this resource

	Data type	Cardinality	Description	
Request body	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.

9.4.10.3.3 PUT

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.10.3.4 PATCH

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.4.10.3.5 DELETE

This method is not supported. When this method is requested on this resource, the OSS/BSS shall return a "405 Method Not Allowed" response as defined in clause 4.3.5.4.

9.5 Data model

9.5.1 Introduction

This clause defines the request and response data structures of the VNF package management interface. If a request or response contains attributes not defined in the present document, 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 data structures to be used in resource representations and notifications.

9.5.2.2 Type: CreateVnfPkgInfoRequest

This type represents the request parameters for creating a new individual VNF package resource. It shall comply with the provisions defined in Table 9.5.2.2-1.

Table 9.5.2.2-1: Definition of the CreateVnfPkgInfoRequest data type

Attribute name	Data type	Cardinality	Description
userDefinedData	KeyValuePairs	0..1	User defined data for the VNF package.

9.5.2.3 Type: VnfPkgInfoModifications

This type represents modifications to the information of a VNF package. It shall comply with the provisions defined in Table 9.5.2.3-1.

Table 9.5.2.3-1: Definition of the VnfPkgInfoModifications data type

Attribute name	Data type	Cardinality	Description
operationalState	PackageOperationalStateType	0..1	New value of the operational state of the on-boarded instance of the VNF package. See note.
userDefinedData	KeyValuePairs	0..1	User defined data to be updated. For existing keys, the value is replaced. See note.
NOTE: At least one of the two parameters shall be present. If the VNF package is not on-boarded, the operation is used only to update existing or add additional user defined data using the userDefinedData attribute.			

9.5.2.4 Type: UploadVnfPackageFromUriRequest

This type represents the request parameters for uploading the content of a VNF package. The NFVO can obtain the VNF package content through the information provided in the request parameters. It shall comply with the provisions defined in Table 9.5.2.4-1.

Table 9.5.2.4-1: Definition of the UploadVnfPackageFromUriRequest data type

Attribute name	Data type	Cardinality	Description
addressInformation	Uri	1	Address information of the VNF package content. The NFVO can use this address to obtain the VNF package.
userName	String	0..1	User name to be used for authentication. Shall be present if user name is needed but has not been provisioned out of band.
password	String	0..1	Password to be used for authentication. Shall be present if password is needed but has not been provisioned out of band.

9.5.2.5 Type: VnfPkgInfo

This type represents the information of a VNF package. It shall comply with the provisions defined in Table 9.5.2.5-1.

Table 9.5.2.5-1: Definition of the VnfPkgInfo data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO.
vnfdId	Identifier	0..1	This identifier, which is managed by the VNF provider, identifies the VNF package and the VNFD in a globally unique way. It is copied from the VNFD of the on-boarded VNF package. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfProvider	String	0..1	Provider of the VNF package and the VNFD. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfProductName	String	0..1	Name to identify the VNF product. Invariant for the VNF product lifetime. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfSoftwareVersion	Version	0..1	Software version of the VNF. This is changed when there is any change to the software included in the VNF package. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
vnfdVersion	Version	0..1	The version of the VNFD. This information is copied from the VNFD. It shall be present after the VNF package content has been on-boarded and absent otherwise.
checksum	Checksum	0..1	Checksum of the on-boarded VNF package. It shall be present after the VNF package content has been on-boarded and absent otherwise.
softwareImages	VnfPackageSoftwareImageInfo	0..N	Information about VNF package artifacts that are software images. This attribute shall not be present before the VNF package content is on-boarded. Otherwise, this attribute shall be present unless it has been requested to be excluded per attribute selector.

Attribute name	Data type	Cardinality	Description
additionalArtifacts	VnfPackageArtifactInfo	0..N	Information about VNF package artifacts contained in the VNF package that are not software images. This attribute shall not be present before the VNF package content is on-boarded. Otherwise, this attribute shall be present if the VNF package contains additional artifacts.
onboardingState	PackageOnboardingStateType	1	On-boarding state of the VNF package. See note 3.
operationalState	PackageOperationalStateType	1	Operational state of the VNF package. See notes 1 and 3.
usageState	PackageUsageStateType	1	Usage state of the VNF package. See notes 2 and 3.
userDefinedData	KeyValuePairs	0..1	User defined data for the VNF package.
_links	Structure (inlined)	1	Links to resources related to this resource.
>self	Link	1	URI of this resource.
>vnfd	Link	0..1	Link to the VNFD resource. This link shall be present after the VNF package content is on-boarded.
>packageContent	Link	1	Link to the "VNF package content" resource.
NOTE 1: If the value of the onboardingState attribute is not equal to "ONBOARDED", the value of the operationalState attribute shall be equal to "DISABLED". NOTE 2: If the value of the onboardingState attribute is not equal to "ONBOARDED", the value of the usageState attribute shall be equal to "NOT_IN_USE". NOTE 3: State changes of a VNF package are illustrated in clause B.2.			

9.5.2.6 Type: PkgmSubscriptionRequest

This type represents a subscription request related to VNF package management notifications about VNF package on-boarding or changes. It shall comply with the provisions defined in Table 9.5.2.6-1.

Table 9.5.2.6-1: Definition of the PkgmSubscriptionRequest data type

Attribute name	Data type	Cardinality	Description
filter	PkgmNotificationsFilter	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.

9.5.2.7 Type: PkgmSubscription

This type represents a subscription related to notifications about VNF package management. It shall comply with the provisions defined in Table 9.5.2.7-1.

Table 9.5.2.7-1: Definition of the PkgmSubscription data type

Attribute name	Data type	Cardinality	Description
id	Identifier	1	Identifier of this subscription resource.
filter	PkgmNotificationsFilter	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.

9.5.2.8 Type: VnfPackageOnboardingNotification

This type represents a VNF package management notification, which informs the receiver that the onboarding process of a VNF package is complete and the package is ready for use. A change of the on-boarding state before the VNF package is on-boarded is not reported. It shall comply with the provisions defined in Table 9.5.2.8-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when the value of the "onboardingState" attribute of a new VNF package has changed to "ONBOARDED".

Table 9.5.2.8-1: Definition of the VnfPackageOnboardingNotification 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 "VnfPackageOnboardingNotification" 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.
vnfPkgId	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO. Its value is the same as the value of the "id" attribute of the related "Individual VNF package" resource.
vnfdId	Identifier	1	This identifier, which is managed by the VNF provider, identifies the VNF package and the VNFD in a globally unique way. It is copied from the VNFD of the on-boarded VNF package.
_links	PkgmLinks	1	Links to resources related to this notification.

9.5.2.9 Type: VnfPackageChangeNotification

This type represents a VNF package management notification, which informs the receiver of a change of the status in an on-boarded VNF package. Only changes in the "operationalState" attribute of an on-boarded VNF package and the deletion of the VNF package will be reported. Change in the "usageState" and "onboardingState" attributes are not reported. The notification shall comply with the provisions defined in Table 9.5.2.9-1. The support of this notification is mandatory. The notification shall be triggered by the NFVO when there is a change in the status of an onboarded VNF package, as follows.

- The "operationalState" attribute of a VNF package has changed, and the "onboardingState" attribute of the package has the value "ONBOARDED".

- The on-boarded VNF package has been deleted.

Table 9.5.2.9-1: Definition of the VnfPackageChangeNotification 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 "VnfPackageChangeNotification" 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.
vnfPkgId	Identifier	1	Identifier of the VNF package. This identifier is allocated by the NFVO. Its value is the same as the value of the "id" attribute of the related "Individual VNF package" resource.
vnfdId	Identifier	1	Identifier of the VNFD contained in the VNF package, which also identifies the VNF package. This identifier is allocated by the VNF provider and copied from the VNFD.
changeType	PackageChangeType	1	The type of change of the VNF package.
operationalState	PackageOperationalStateType	0..1	New operational state of the VNF package. Only present when changeType is OP_STATE_CHANGE.
_links	PkgmLinks	1	Links to resources related to this notification.

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 are not resource representations.

9.5.3.2 Type: VnfPackageSoftwareImageInfo

This type represents an artifact contained in a VNF package which represents a software image. It shall comply with provisions defined in Table 9.5.3.2-1.

Table 9.5.3.2-1: Definition of the VnfPackageSoftwareImageInfo data type

Attribute name	Data type	Cardinality	Description
id	IdentifierInVnfd	1	Identifier of the software image.
name	String	1	Name of the software image.
provider	String	1	Provider of the software image.
version	Version	1	Version of the software image.
checksum	Checksum	1	Checksum of the software image file.
containerFormat	Enum (inlined)	1	Container format indicates whether the software image is in a file format that also contains metadata about the actual software. Permitted values: - AKI: a kernel image format - AMI: a machine image format - ARI: a ramdisk image format - BARE: the image does not have a container or metadata envelope - DOCKER: docker container format - OVA: OVF package in a tarfile - OVF: OVF container format See note 1.

Attribute name	Data type	Cardinality	Description
diskFormat	Enum (inlined)	1	Disk format of a software image is the format of the underlying disk image. Permitted values: - AKI: a kernel image format - AMI: a machine image format - ARI: a ramdisk image format - ISO: an archive format for the data contents of an optical disc, such as CD-ROM - QCOW2: a common disk image format, which can expand dynamically and supports copy on write - RAW: an unstructured disk image format - VDI: a common disk image format - VHD: a common disk image format - VHDX: enhanced version of VHD format - VMDK: a common disk image format See note 2.
createdAt	DateTime	1	Time when this software image was created.
minDisk	UnsignedInt	1	The minimal disk for this software image in bytes.
minRam	UnsignedInt	1	The minimal RAM for this software image in bytes.
size	UnsignedInt	1	Size of this software image in bytes.
userMetadata	KeyValuePairs	0..1	User-defined data.
imagePath	String	1	Path in the VNF package, which identifies the image artifact and also allows to access a copy of the image artifact.
NOTE 1: The list of permitted values was taken from "Container formats" in [i.4]. NOTE 2: The list of permitted values was adapted from "Disk formats" in [i.4].			

9.5.3.3 Type: VnfPackageArtifactInfo

This type represents an artifact other than a software image which is contained in a VNF package. It shall comply with provisions defined in Table 9.5.3.3-1.

Table 9.5.3.3-1: Definition of the VnfPackageArtifactInfo data type

Attribute name	Data type	Cardinality	Description
artifactPath	String	1	Path in the VNF package, which identifies the artifact and also allows to access a copy of the artifact.
checksum	Checksum	1	Checksum of the artifact file.
metadata	KeyValuePairs	0..1	The metadata of the artifact that are available in the VNF package, such as Content type, size, creation date, etc.

9.5.3.4 Type: PkgmNotificationsFilter

This type represents a subscription filter related to notifications related to VNF package management. It shall comply with the provisions defined in Table 9.5.3.4-1.

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 9.5.3.4-1: Definition of the PkgmNotificationsFilter data type

Attribute name	Data type	Cardinality	Description
notificationTypes	Enum (inlined)	0..N	Match particular notification types. Permitted values: - VnfPackageOnboardingNotification - VnfPackageChangeNotification See note 1.
vnfProductsFromProviders	Structure (inlined)	0..N	If present, match VNF packages that contain VNF products from certain providers. See note 2.
>vnfProvider	String	1	Name of the VNFprovider to match.
>vnfProducts	Structure (inlined)	0..N	If present, match VNF packages that contain 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 packages that contain VNF products with certain versions and a certain product name, from one particular provider.
>>>vnfSoftwareVersion	Version	1	VNF software version to match.
>>>vnfdVersions	Version	0..N	If present, match VNF packages that contain VNF products with certain VNFD versions, a certain software version and a certain product name, from one particular provider.
vnfdId	Identifier	0..N	Match VNF packages with a VNFD identifier listed in the attribute. See note 2.
vnfPkgId	Identifier	0..N	Match VNF packages with a package identifier listed in the attribute. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise. See note 2.
operationalState	PackageOperationalStateType	0..N	Match particular operational state of the VNF package. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise.
usageState	PackageUsageStateType	0..N	Match particular usage state of the VNF package. May be present if the "notificationTypes" attribute contains the value "VnfPackageChangeNotification", and shall be absent otherwise.
NOTE 1: The permitted values of the "notificationTypes" attribute are spelled exactly as the names of the notification types to facilitate automated code generation systems. NOTE 2: The attributes "vnfProductsFromProviders", "vnfdId", and "vnfPkgId" are alternatives to reference particular VNF packages in a filter. They should not be used both in the same filter instance, but one alternative should be chosen.			

9.5.3.5 Type: PkgmLinks

This type represents the links to resources that a VNF package management notification can contain. It shall comply with the provisions defined in Table 9.5.3.5-1.

Table 9.5.3.5-1: Definition of the PkgmLinks data type

Attribute name	Data type	Cardinality	Description
vnfPackage	NotificationLink	1	Link to the resource representing the VNF package to which the notified change applies, i.e. the individual on-boarded VNF package resource that represents the VNF package.
subscription	NotificationLink	1	Link to the related subscription.

9.5.3.6 Type: Checksum

This type represents the checksum of a VNF package or an artifact file. It shall comply with the provisions defined in Table 9.5.3.6-1.

Table 9.5.3.6-1: Definition of the Checksum data type

Attribute name	Data type	Cardinality	Description
algorithm	String	1	Name of the algorithm used to generate the checksum, as defined in ETSI GS NFV-SOL 004 [5]. For example, SHA-256, SHA-512.
hash	String	1	The hexadecimal value of the checksum.

9.5.4 Referenced simple data types and enumerations

9.5.4.1 Introduction

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

9.5.4.2 Simple data types

No particular simple data types are defined for this interface, in addition to those defined in clause 4.4.2.

9.5.4.3 Enumeration: PackageOnboardingStateType

The enumeration PackageOnboardingStateType shall comply with the provisions defined in Table 9.5.4.3-1.

Table 9.5.4.3-1: Enumeration PackageOnboardingStateType

Enumeration value	Description
CREATED	The VNF package resource has been created.
UPLOADING	The associated VNF package content is being uploaded.
PROCESSING	The associated VNF package content is being processed, e.g. validation.
ONBOARDED	The associated VNF package content is successfully on-boarded.

9.5.4.4 Enumeration: PackageOperationalStateType

The enumeration PackageOperationalStateType shall comply with the provisions defined in Table 9.5.4.4-1.

Table 9.5.4.4-1: Enumeration PackageOperationalStateType

Enumeration value	Description
ENABLED	The VNF package is enabled, i.e. it can be used for instantiation of new VNF instances.
DISABLED	The VNF package is disabled, i.e. it cannot be used for further VNF instantiation requests (unless and until the VNF package is re-enabled).

9.5.4.5 Enumeration: PackageUsageStateType

The enumeration PackageUsageStateType shall comply with the provisions defined in Table 9.5.4.5-1.

Table 9.5.4.5-1: Enumeration PackageUsageStateType

Enumeration value	Description
IN_USE	VNF instances instantiated from this VNF package exist.
NOT_IN_USE	No existing VNF instance is instantiated from this VNF package.

9.5.4.6 Enumeration: PackageChangeType

The enumeration PackageChangeType shall comply with the provisions defined in Table 9.5.4.6-1.

Table 9.5.4.6-1: Enumeration PackageChangeType

Enumeration value	Description
OP_STATE_CHANGE	The "operationalState" attribute has been changed.
PKG_DELETE	The VNF package has been deleted.

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 013 [3] and the corresponding resources and HTTP methods defined in the present document.

A.2 NSD Management interface

The mapping of NSD management interface operations, defined in ETSI GS NFV-IFA 013 [3], to the resources and HTTP methods defined in the present document can be found in Table A.2-1.

Table A.2-1: Mapping of ETSI GS NFV-IFA0 13 [3] NSD Management interface operations with resources and HTTP methods

ETSI GS NFV-IFA 013 [3] NSD Management interface operation	HTTP method	Resource	Direction
Create NSD Info	POST	nsd/v1/ns_descriptors	OSS/BSS → NFVO
Upload NSD	PUT	nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content	OSS/BSS → NFVO
Update NSD Info	PATCH	nsd/v1/ns_descriptors/{nsdInfoId}	OSS/BSS → NFVO
Delete NSD	DELETE	nsd/v1/ns_descriptors/{nsdInfoId}	OSS/BSS → NFVO
Query NSD	GET	nsd/v1/ns_descriptors	OSS/BSS → NFVO
	GET	nsd/v1/ns_descriptors/{nsdInfoId}	OSS/BSS → NFVO
Fetch NSD	GET	nsd/v1/ns_descriptors/{nsdInfoId}/nsd_content	OSS/BSS → NFVO
Create PNFD Info	POST	nsd/v1/pnf_descriptors	OSS/BSS → NFVO
Upload PNFD	PUT	nsd/v1/pnf_descriptors/{pnfdInfoId}/pnfd_content	OSS/BSS → NFVO
Update PNFD Info	POST	nsd/v1/pnf_descriptors/{pnfdInfoId}	OSS/BSS → NFVO
Delete PNFD	DELETE	nsd/v1/pnf_descriptors/{pnfdInfoId}	OSS/BSS → NFVO
Query PNFD Info	GET	nsd/v1/pnf_descriptors	OSS/BSS → NFVO
	GET	nsd/v1/pnf_descriptors/{pnfdInfoId}	OSS/BSS → NFVO
Fetch PNFD	GET	nsd/v1/pnf_descriptors/{pnfdInfoId}/pnfd_content	OSS/BSS → NFVO
Subscribe	POST	nsd/v1/subscriptions	OSS/BSS → NFVO
Query Subscription Information	GET	nsd/v1/subscriptions	OSS/BSS → NFVO
	GET	nsd/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Terminate Subscription	DELETE	nsd/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Notify	POST	(client-provided)	NFVO → OSS/BSS

A.3 NS Lifecycle Management interface

The mapping of NS lifecycle management operations, defined in ETSI GS NFV-IFA 013 [3], to the resources and HTTP methods defined in the present document can be found in Table A.3-1.

Table A.3-1: Mapping of ETSI GS NFV-IFA 013 [3] operations with NS Lifecycle Management interface resources and methods

ETSI GS NFV-IFA 013 [3] NS Lifecycle Management interface operation	HTTP method	Resource	Direction
Create NS Identifier	POST	nslcm/v1/ns_instances	OSS/BSS → NFVO
Instantiate NS	POST	nslcm/v1/ns_instances/{nsInstanceId}/instantiate	OSS/BSS → NFVO
Scale NS	POST	nslcm/v1/ns_instances/{nsInstanceId}/scale	OSS/BSS → NFVO
Update NS	POST	nslcm/v1/ns_instances/{nsInstanceId}/update	OSS/BSS → NFVO
Terminate NS	POST	nslcm/v1/ns_instances/{nsfInstanceId}/terminate	OSS/BSS → NFVO

ETSI GS NFV-IFA 013 [3] NS Lifecycle Management interface operation	HTTP method	Resource	Direction
Delete NS Identifier	DELETE	nslcm/v1/ns_instances/{nsInstanceId}	OSS/BSS → NFVO
Query NS	GET	nslcm/v1/ns_instances/{nsInstanceId}	OSS/BSS → NFVO
	GET	nslcm/v1/ns_instances	OSS/BSS → NFVO
Heal NS	POST	nslcm/v1/ns_instances/{nsInstanceId}/heal	OSS/BSS → NFVO
Get Operation Status	GET	nslcm/v1/ns_lcm_op_occs	OSS/BSS → NFVO
	GET	nslcm/v1/ns_lcm_op_occs/{nsLcmOpOccId}	OSS/BSS → NFVO
Subscribe	POST	nslcm/v1/subscriptions	OSS/BSS → NFVO
Query Subscription Information	GET	nslcm/v1/subscriptions	OSS/BSS → NFVO
	GET	nslcm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Terminate Subscription	DELETE	nslcm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Notify	POST	(client-provided)	NFVO → OSS/BSS

A.4 NS Performance Management interface

The mapping of NS performance management operations, defined in ETSI GS NFV-IFA 013 [3], to the resources and HTTP methods defined in the present document can be found in Table A.4-1.

Table A.4-1: Mapping of ETSI GS NFV-IFA 013 [3] NS Performance Management interface operations with resources and HTTP methods

ETSI GS NFV-IFA013 [3] NS Performance Management operation	HTTP method	Resource	Direction
Create PM Job	POST	nspm/v1/pm_jobs	OSS/BSS → NFVO
Delete PM Job	DELETE	nspm/v1/pm_jobs/{pmJobId}	OSS/BSS → NFVO
Query PM Job	GET	nspm/v1/pm_jobs	OSS/BSS → NFVO
	GET	nspm/v1/pm_jobs/{pmJobId}	OSS/BSS → NFVO
Create Threshold	POST	nspm/v1/thresholds	OSS/BSS → NFVO
Delete Threshold	DELETE	nspm/v1/thresholds/{thresholdId}	OSS/BSS → NFVO
Query Threshold	GET	nspm/v1/thresholds	OSS/BSS → NFVO
	GET	nspm/v1/thresholds/{thresholdId}	OSS/BSS → NFVO
Subscribe	POST	nspm/v1/subscriptions	OSS/BSS → NFVO
Query Subscription Information	GET	nspm/v1/subscriptions	OSS/BSS → NFVO
	GET	nspm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Terminate Subscription	DELETE	nspm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Notify	POST	(client-provided)	NFVO → OSS/BSS

A.5 NS Fault Management interface

The mapping of NS fault management operations, defined in ETSI GS NFV-IFA 013 [3], to the resources and HTTP methods defined in the present document can be found in Table A.5-1.

Table A.5-1: Mapping of ETSI GS NFV-IFA 013 [3] NS Fault Management interface operations with resources and HTTP methods

ETSI GS NFV-IFA 013 [3] NS Fault Management interface operation	HTTP method	Resource	Direction
Get Alarm List	GET	nsfm/v1/alarms	OSS/BSS → NFVO
	GET	nsfm/v1//alarms/{alarmId}	OSS/BSS → NFVO
Acknowledge Alarm	PATCH	nsfm/v1//alarms/{alarmId}	OSS/BSS → NFVO
Subscribe	POST	nsfm/v1/subscriptions	OSS/BSS → NFVO
Query Subscription Information	GET	nsfm/v1/subscriptions	OSS/BSS → NFVO
	GET	nsfm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Terminate Subscription	DELETE	nsfm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Notify	POST	(client-provided)	NFVO → OSS/BSS

A.6 VNF Package Management interface

The mapping of VNF package management operations, defined in ETSI GS NFV-IFA 013 [3], to the resources and HTTP methods defined in the present document can be found in Table A.6-1.

Table A.6-1: Mapping of ETSI GS NFV-IFA 013 [3] operations with VNF Package Management interface resources and methods

ETSI GS NFV-IFA 013 [3] VNF Package Management interface operation	HTTP method	Resource	Direction
Create VNF Package Info	POST	vnfpkgm/v1/vnf_packages	OSS/BSS → NFVO
Update VNF Package Info	PATCH	vnfpkgm/v1/vnf_packages/{vnfPkgId}	OSS/BSS → NFVO
Delete VNF Package	DELETE	vnfpkgm/v1/vnf_packages/{vnfPkgId}	OSS/BSS → NFVO
Query VNF Package Info	GET	vnfpkgm/v1/vnf_packages	OSS/BSS → NFVO
	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}	OSS/BSS → NFVO
	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/vnfd	OSS/BSS → NFVO
Upload VNF Package	PUT	vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content	OSS/BSS → NFVO
	POST	vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content/upload_from_uri	OSS/BSS → NFVO
Fetch VNF Package	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/package_content	OSS/BSS → NFVO
Fetch VNF Package Artifacts	GET	vnfpkgm/v1/vnf_packages/{vnfPkgId}/artifacts/{artifactPath}	OSS/BSS → NFVO
Subscribe	POST	vnfpkgm/v1/subscriptions	OSS/BSS → NFVO
Query Subscription Information	GET	vnfpkgm/v1/subscriptions	OSS/BSS → NFVO
	GET	vnfpkgm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Terminate subscription	DELETE	vnfpkgm/v1/subscriptions/{subscriptionId}	OSS/BSS → NFVO
Notify	POST	(client-provided)	NFVO → OSS/BSS

Annex B (informative): State models

B.1 NSD state model

B.1.1 Introduction

This clause describes the state model of NSD in the NFVO. It includes the state models for two phases, i.e. onboarding phase of NSD and operational phase of NSD.

B.1.2 State model

A given NSD has three states, i.e. on-boarding state, operational state and usage state.

The on-boarding state is represented by the "nsdOnboardingState" attribute in the "NsdInfo" data type with below values:

- CREATED: The NSD information object is created.
- UPLOADING: The NSD is being uploaded.
- PROCESSING: The NSD is being processed, e.g. validation.
- ONBOARDED: The NSD is successfully on-boarded.

The operational state is represented by the "nsdOperationalState" attribute in the "NsdInfo" data type with below values:

- ENABLED: The NSD is enabled.
- DISABLED: The NSD is disabled.

The usage state is represented by the "nsdUsageState" attribute in the "NsdInfo" data type with below values:

- IN_USE: The NSD is in use.
- NOT_IN_USE: The NSD is not in use.

The state model of on-boarding phase in Figure B.1.2-1 applies to a given NSD being on-boarded. Besides the operations and conditions specified in the Figure, below operations are also considered as available during the on-boarding phase:

- Query NSD Info
- Update NSD Info (with user defined data only)

The state model of operational phase in Figure B.1.2-1 applies to an on-boarded NSD. Besides the operations and conditions specified in the Figure, below operations are also considered as available during the operational phase:

- Query NSD Info
- Update NSD Info (with user defined data only)
- Fetch NSD

At the end of the on-boarding phase, the "nsdOnboardingState" value transitions to "ONBOARDED" and the "nsdOperationalState" value transitions from "DISABLED" to "ENABLED", and the operational phase is entered.

The "nsdOperationalState" and "nsdUsageState" detail the state changes during the NSD operational phase. During the NSD on-boarding phase, the value of the "nsdOperationalState" is "DISABLED" and the value of the "nsdUsageState" is "NOT_ IN_USE". Right after the NSD becomes on-boarded, the value of the "nsdOperationalState" is changed to "ENABLED" and the value of the "nsdUsageState" is kept as "NOT_ IN_USE".

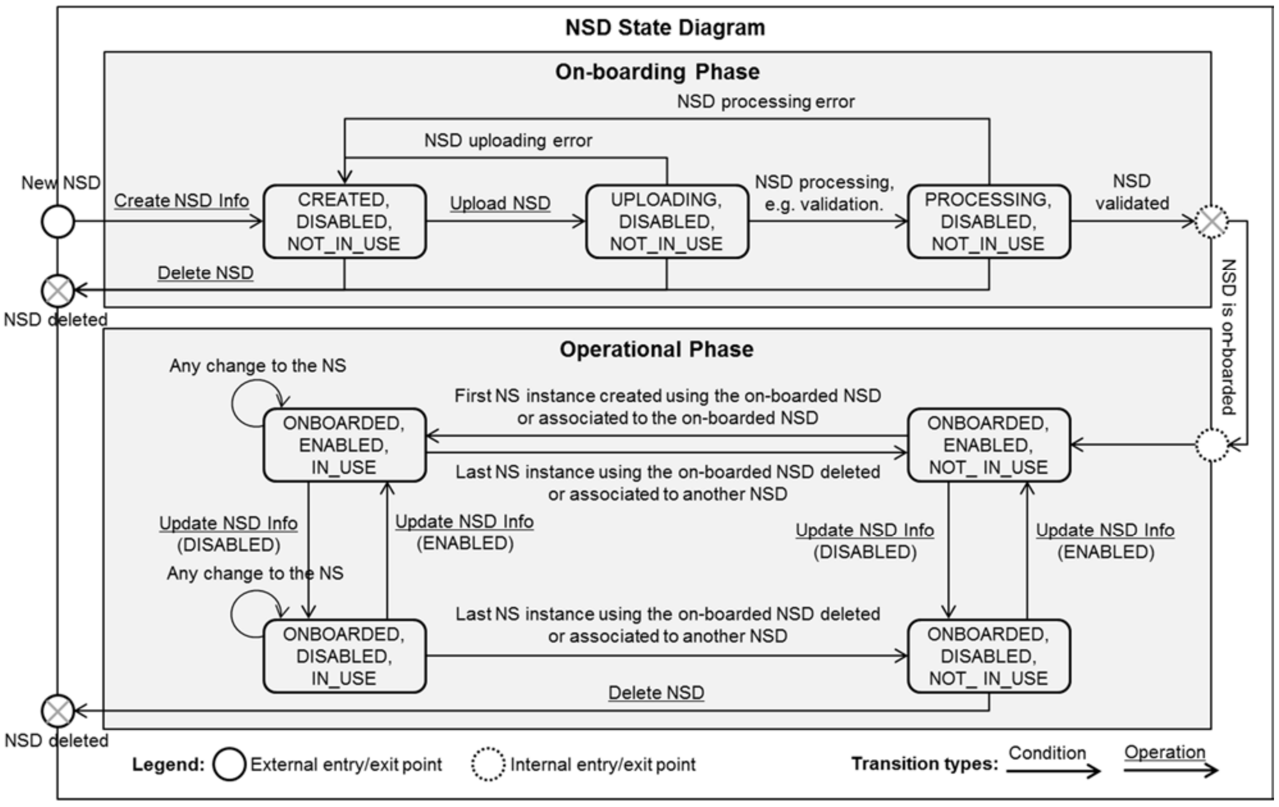
NSD State Diagram
On-boarding Phase
NSD processing error
NSD uploading error
New NSD
Create NSD Info
CREATED, DISABLED, NOT_IN_USE
Upload NSD
UPLOADING, DISABLED, NOT_IN_USE
NSD processing, e.g. validation.
PROCESSING, DISABLED, NOT_IN_USE
NSD validated
Delete NSD
NSD deleted
NSD is on-boarded
Operational Phase
Any change to the NS
First NS instance created using the on-boarded NSD or associated to the on-boarded NSD
ONBOARDED, ENABLED, IN_USE
ONBOARDED, ENABLED, NOT_ IN_USE
Last NS instance using the on-boarded NSD deleted or associated to another NSD
Update NSD Info (DISABLED)
Update NSD Info (ENABLED)
Update NSD Info (DISABLED)
Update NSD Info (ENABLED)
Any change to the NS
ONBOARDED, DISABLED, IN_USE
Last NS instance using the on-boarded NSD deleted or associated to another NSD
ONBOARDED, DISABLED, NOT_ IN_USE
Delete NSD
NSD deleted
Legend: External entry/exit point
Internal entry/exit point
Transition types: Condition
Operation

Figure B.1.2-1: NSD state model

B.2 VNF package state model

B.2.1 Introduction

This clause describes the state model of VNF Package in the NFVO. It includes the state models for two phases, i.e. on-boarding phase and operational phase.

B.2.2 State model

A given VNF Package has three states, i.e. on-boarding state, operational state and usage state.

The on-boarding state is represented by the "onboardingState" attribute in the "VnfPkgInfo" information element with below values:

- CREATED: The VNF Package information object is created.
- UPLOADING: The VNF Package is being uploaded.
- PROCESSING: The VNF Package is being processed, e.g. validation.
- ONBOARDED: The VNF Package is successfully on-boarded.

The operational state is represented by the "operationalState" attribute in the "VnfPkgInfo" information element with below values:

- ENABLED: The VNF Package is enabled.
- DISABLED: The VNF Package is disabled.

The usage state is represented by the "usageState" attribute in the "VnfPkgInfo" information element with below values:

- IN_USE: The VNF Package is in use.
- NOT_IN_USE: The VNF Package is not in use.

The state model of on-boarding phase in Figure B.2.2-1 applies to a given VNF Package being on-boarded. Besides the operations and conditions specified in the Figure, below operations are also considered as available during the on-boarding phase:

- Query VNF Package Info
- Update VNF Package Info (with user defined data only)

The state model of operational phase in Figure B.2.2-1 applies to an on-boarded VNF Package. Besides the operations and conditions specified in the Figure, below operations are also considered as available during the operational phase:

- Query VNF Package Info
- Update VNF Package Info (with user defined data only)
- Fetch VNF Package
- Fetch VNF Package Artifacts

The "onboardingState" details the state changes during the VNF Package on-boarding phase. The value of this attribute during the VNF Package operational phase is "ONBOARDED".

The "operationalState" and "usageState" detail the state changes during the VNF Package operational phase. During the VNF Package on-boarding phase, the value of the "operationalState" is "DISABLED" and the value of the "usageState" is "NOT_ IN_USE". Right after the VNF Package becomes on-boarded, the value of the "operationalState" is changed to "ENABLED" and the value of the "usageState" is kept as "NOT_ IN_USE", as shown in Figure B.2.2-1.

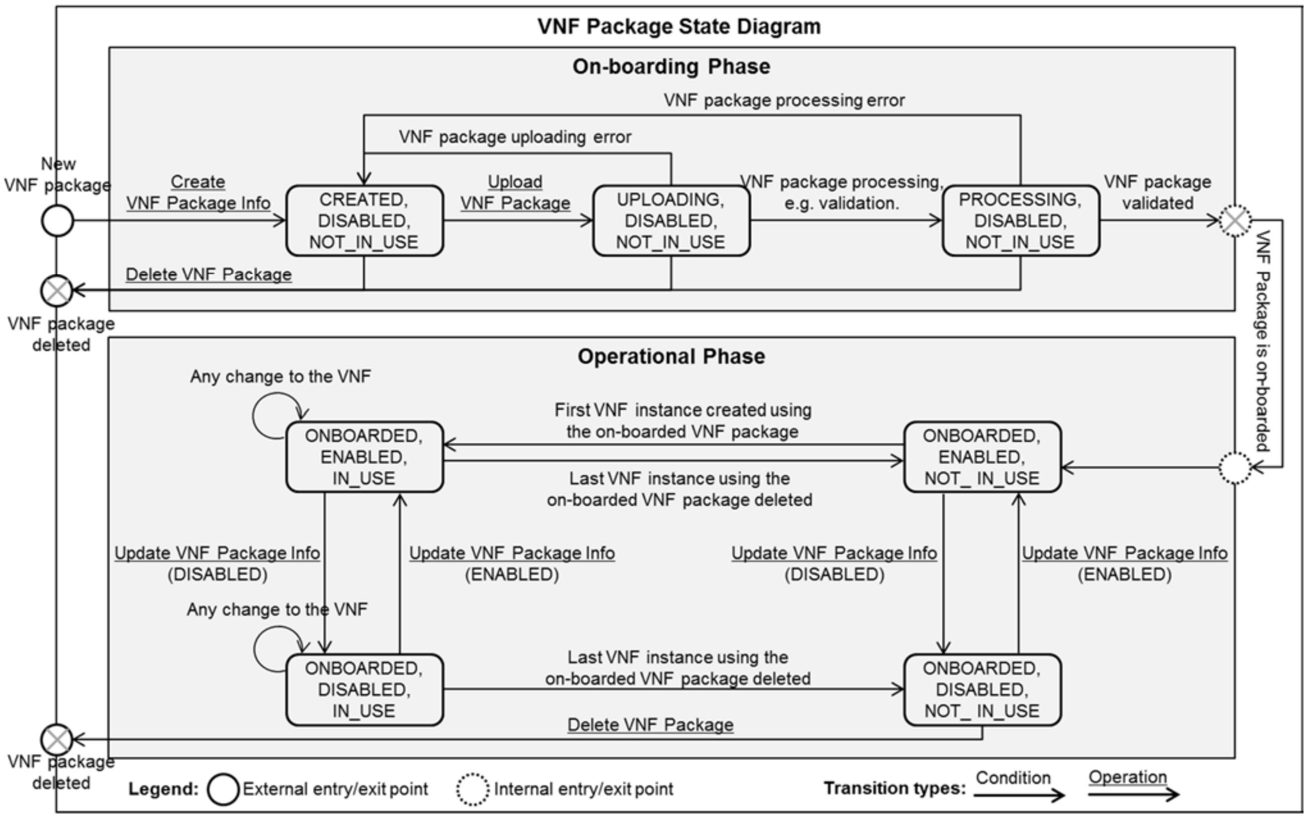
VNF Package State Diagram
On-boarding Phase
VNF package processing error
VNF package uploading error
New VNF package
Create VNF Package Info
CREATED, DISABLED, NOT_IN_USE
Upload VNF Package
UPLOADING, DISABLED, NOT_IN_USE
VNF package processing, e.g. validation.
PROCESSING, DISABLED, NOT_IN_USE
VNF package validated
Delete VNF Package
VNF package deleted
VNF Package is on-boarded
Operational Phase
Any change to the VNF
ONBOARDED, ENABLED, IN_USE
First VNF instance created using the on-boarded VNF package
ONBOARDED, ENABLED, NOT_ IN_USE
Last VNF instance using the on-boarded VNF package deleted
Update VNF Package Info (DISABLED)
Update VNF Package Info (ENABLED)
Update VNF Package Info (DISABLED)
Update VNF Package Info (ENABLED)
Any change to the VNF
ONBOARDED, DISABLED, IN_USE
Last VNF instance using the on-boarded VNF package deleted
ONBOARDED, DISABLED, NOT_ IN_USE
Delete VNF Package
VNF package deleted
Legend: External entry/exit point
Internal entry/exit point
Transition types: Condition
Operation

Figure B.2.2-1: VNF Package state model

Annex C (informative): Complementary material for API utilization

To complement the definitions of each method, resource, and data type defined in the main body of the present document, the ETSI NFV ISG is providing supplementary description files, compliant to the OpenAPI Specification [i.5], for the Os-Ma-nfvo reference point. These supplementary description files, containing the OpenAPI specification for each API defined in the present document, are located at https://forge.etsi.org/rep/nfv/NFV-SOL005.

In case of discrepancies between the supplementary files and the related data structure definitions in the main body of the present document, the data structure definitions take precedence.

The OpenAPI representations referenced above:

1) use the MAJOR.MINOR.PATCH version fields to signal the version of the API as defined in the present document; and

2) use the "impl" version parameter to represent changes to the OpenAPI representation without changing the present document (see clause 4.6.1.2).

The full version identifier of an API appears in the corresponding OpenAPI file, in the "version" subfield of the "info" field, where numerical fields are separated by a dot, as illustrated below.

EXAMPLE:

```
swagger: "2.0"
info:
  version: "1.0.0-impl:etsi.org:ETSI_NFV_OpenAPI:1"
  title: SOL005 NS LCM
  license:
    name: "ETSI Forge copyright notice"
    url: https://forge.etsi.org/etsi-forge-copyright-notice.txt
basePath: "/nslcm/v1"
```

END EXAMPLE

Annex D (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Ernest Bayha, Ericsson LM

Other contributors:

Hidenori Asaba, DOCOMO Communications Lab

Bruno Chatras, ORANGE

Haibin Chu, Ericsson LM

Gang He, China Unicom

Manchang Ju, ZTE

Yuya Kuno, DOCOMO Communications Lab

Jie Miao, China Unicom

Jiaqiang Pan, ZTE

Anne-Marie Praden, Gemalto N.V.

Uwe Rauschenbach, Nokia Networks

Kazi Wali Ullah, Ericsson LM

Xu Yang, Huawei Technologies Co. Ltd.

Jong-Hwa Yi, ETRI

Yaoye Zhang, Huawei Technologies Co. Ltd.

Peng Zhao, China Mobile

Annex E (informative): Change History

Date	Version	Information about changes
December 2016	0.0.1	Initial version based on contributions that were agreed at the NFVSOL#15 meeting - NFVSOL(16)000169 SOL005_Scope_Statement - NFVSOL(16)000171_SOL005_Proposed_Table_of_Contents
February 2017	0.1.0	Version 0.1.0 based on contributions that were agreed at the NFVSOL#20 meeting - NFVSOL(17)000061_SOL005_URI_structure_and_supported_content_formats - NFVSOL(17)000064R1_SOL002_SOL003_SOL005_Labeling_of_API_names - NFVSOL(17)000095R1_SOL005_Error_reporting - NFVSOL(17)000106R1_Conventions_document_NFVSOL(17)000050_swagger_representation_of_the_API - NFVSOL(17)000107R1_SOL005_Common_procedures - NFVSOL(17)000111_SOL003_Conventions_move_Resource_structure_up_in_the_TOC
March 2017	0.2.0	Version 0.2.0 based on contributions that were agreed at the NFVSOL#22 meeting - NFVSOL(17)000196R1_SOL005_Add_SOL003_Normative_Reference - NFVSOL(17)000198_SOL005_Clause_4.1_Overview - NFVSOL(17)000199_SOL002_SOL003_Renaming_attribute_selectors - NFVSOL(17)000200_SOL002_SOL003_Attribute_filter_equality - NFVSOL(17)000179_SOL005_Simple_Data_Types - NFVSOL(17)000138_SOL005_NS_LCM_Description_Clause_6.1 - NFVSOL(17)000142R1_SOL005_NS_LCM_Resource_Structure_and_Methods_Clause_6.2 - NFVSOL(17)000123R2_SOL005_Flow_of_the_Creation_of_a_NS_Instance_Resource - NFVSOL(17)000129R1_SOL005_Flow_of_the_Deletion_of_a_NS_Instance_Resource - NFVSOL(17)000130_SOL005_Flow_of_the_Get_Operations_Status_Operation - NFVSOL(17)000137_SOL005_Flow_of_Managing_Subscriptions_to_Notifications_Related_to_NS_Lifecycle_Management - NFVSOL(17)000136_SOL005_Flow_of_Sending_Notifications_Related_to_NS_Lifecycle_management - NFVSOL(17)000156R2_SOL005_NS_Lifecycle_Change_Resource_Definitions_Methods_and_Data_Types - NFVSOL(17)000160_SOL005_NS_Lifecycle_Management_Resource_Definitions_Methods_and_Data_Types
April 2017	0.3.0	Version 0.3.0 based on contributions that were agreed at the NFVSOL#26 meeting - NFVSOL(17)000263_SOL005_Instantiate_NS_Lifecycle_Management Resource_Definition_Methods_and_Data_Types - NFVSOL(17)000265_SOL005_Heal_NS_Lifecycle_Management Resource_Definition_Methods_and_Data_Types
May 2017	0.4.0	Version 0.4.0 based on contributions that were agreed at the NFVSOL#28 meeting - NFVSOL(17)000264R1_SOL005_Update_NS_Lifecycle_Management_Resource_Methods_and_Data_Types - NFVSOL(17)000266_SOL005_Scale_NS_Operation_Resource_Definition_Methods_and_Data_Types - NFVSOL(17)000267_SOL005_Terminate_NS_Operation_Resource_Definition_Methods_and_Data Type - NFVSOL(17)000268_SOL005_Merge_NS_LCCN_interface_into_the_NS_LCM_interface - NFVSOL(17)000348R4_SOL005_NSD_Management_Interface - NFVSOL(17)000378_SOL005_NS_Lifecycle_Change_Occurrence_Resource_Definitions_Methods_and_Data_Types - NFVSOL(17)000383_SOL005_Flow_of_NS_Lifecycle_Management_Operations_Triggered_by_Task_Resources - NFVSOL(17)000385R1_SOL005_Individual_NS_Descriptor_Resource_Descriptor_Methods_and_Data_Types - NFVSOL(17)000390R1_SOL005_Handling_of_Errors_During_NS_Lifecycle_Management_Operations
June 2017	0.5.0	Version 0.5.0 based on contributions that were agreed at the NFVSOL#30 and NFVSOL#31 meetings - NFVSOL(17)000371R1_SOL005:_Data_Type_NsInstance - NFVSOL(17)000411R1_SOL005:_Clause_6.6_Update,_"Handling_of_Errors_During_NS_Lifecycle_Management"

Date	Version	Information about changes
		- NFVSOL(17)000420R1_SOL005:_Update_ScaleVnfData_and_ScaleByStepData_Data_Types_consistent_with_NFVIFA(17)000382R1 - NFVSOL(17)000421R1_SOL005:_Update_"additionalParam…"_attribute_names, KeyValuePair data type_and_its_cardinality_consistent_with_SOL003 - NFVSOL(17)000422_SOL005:_Update_AffinityOrAntiAffinityRule_Data_Type_Consistent_with_NFVIFA(17)000534 - NFVSOL(17)000424_SOL005:_Modify_VNF_Configuration_in_Update_NS_operation_consistent_with_NFVIFA(17)000527 - NFVSOL(17)000427R2_SOL005:_Update_NSD_Interface_Resource_Tree,_Resources,_and_Methods_(Clause_5.2) - NFVSOL(17)000428_SOL005:_ParamsForVnf_and_Resolution_of_the_Associated_Editor's_Note - NFVSOL(17)000431R3_SOL005:_Error_Handling_for_NS_Lifecycle_operation,_Resource_Definition_and_Methods - NFVSOL(17)000432R2_SOL005:_Update_Flow_of_Error_Handling_for_NS_LCM_Operations - NFVSOL(17)000433R3_SOL005:_Update_"address"_attribute's_ data_type_in_PnfExtCpInfo_data_type - NFVSOL(17)000434R3_SOL005:_Update_LocationConstraints_data type_in_Instantiate_NS_operation - NFVSOL(17)000436R2_SOL005:_Filter_design_for_NS_Instances - NFVSOL(17)000438R2_SOL005_VNF_Package_management_interface_-_resource_structure_and_methods - NFVSOL(17)000439R3_SOL005_VNF_package_management_interface_-_resources - NFVSOL(17)000440R3_SOL005:_Updates_to_the_Lifecycle_Change_Notifications_Filter - NFVSOL(17)000441R1_SOL005:_Authorizations_of_API_requests_and_notifications - NFVSOL(17)000446R1_SOL005:_Update_Clause_4.3_"Common_Procedures"_Consistent_with_SOL003 - NFVSOL(17)000451_SOL005:_Correct_various_references_to_NS_Lifecycle_operation_occurrences - NFVSOL(17)000464_SOL005_-_Modifications_on_sequence_diagrams_in clauses_5.3.2_and 5.3.3 - NFVSOL(17)000466_SOL005:_Add_IdentifierInVim_simple_data_type
August 2017	0.6.0	Version 0.6.0 based on contributions that were agreed at the NFVSOL#34 and NFVSOL#35,meetings - NFVSOL(17)00426R6_SOL005:_Changes_to_NSD_Resources,_Methods, and_Data_Types - NFVSOL(17)00493_SOL005: Update_Clause_3,_Abbreviations - NFVSOL(17)000494R1_SOL005:_NS_PM_Interface_description,_resource_structure,_and_methods - NFVSOL(17)000508_SOL005:_Remove_Editor's Npte_in Clause_6.5.3.8 - NFVSOL(17)000509_SOL005:_Remove_Editor's_Note_in_Clauses 5.5.3.1, 5.5.4.1, and 5.5.4.2 - NFVSOL(17)000510R1_SOL005:_NS_Performance_Management_Interface_resources_and_data_model - NFVSOL(17)000511R1_SOL005:_VNF_package_managment_interface_flows - NFVSOL(17)000512R1_SOL005:_VNF_package_management_interface_data_types - NFVSOL(17)000513R2_SOL005:_NSD_Management_Interface_operations,_resource_structure,_and_methods - NFVSOL(17)000514R1_SOL005:_NS_LCM_Updates
September 2017	0.7.0	Version 0.7.0 based on contributions that were agreed at the NFVSOL#36 and NFVSOL#37 meetings - NFVSOL(17)000386R3_SOL005:_PNFD_resources,_methods,_and_data_types - NFVSOL(17)00478R1_SOL005:_VL_and_CP_consistency_(mirror_of_403r3_and_423) - NFVSOL(17)000519R4_SOL005:_NS_fault_management_interface - NFVSOL(17)000520R1_SOL005:_New_Annex_B_Mapping_operations_to_protocol_elements - NFVSOL(17)000524R1_SOL005:_Refactoring_of_NSD_management_interface - NFVSOL(17)000526R1_SOL005:_Definition of_the_HealNsData_data type (clause_6.5.3.33) - NFVSOL(17)000529_SOL005:_Apply_the_agreed_design_for_VNF_package_on-boarding_operation

Date	Version	Information about changes
		- NFVSOL(17)000530R1_SOL005:_Remaining_VNF_package_management_interface flows - NFVSOL(17)000531R1_SOL005:_Remaining_resource_design_for_VNF_package_management_interface - NFVSOL(17)000532R1_SOL005:_Filters_and_selectors_for_VNF_package_management_interface - NFVSOL(17)000533R2_SOL005:_Additional_data_model_for_VNF_package_management_interface - NFVSOL(17)000534_SOL005:_VNF_package_management_interface_data_type_align_with_SOL003 - NFVSOL(17)000554_SOL005: Align_with_IFA013_on vimId
October 2017	0.8.0	Version 0.8.0 based on contributions that were agreed at the NFVSOL#38 meeting. - NFVSOL(17)000521_SOL005:_NS_performance_management_interface_sequence diagrams - NFVSOL(17)000558R1_SOL005: NSD_management_interface_-_notification_and_state_diagram - NFVSOL(17)000573_SOL005:_NSD_management_interface_subscription_resources
November 2017	0.9.0	Version 0.9.0 based on contributions that were agreed at the NFVSOL#40 and NFVSOL#41 meetings. - NFVSOL(17)000535r2_SOL005:_Modification_of_data_types_due_to_the_separation_of_on-boarding_VNF_package_operation - NFVSOL(17)000546R3_SOL005:_Define_the_nfpRule_attribute - NFVSOL(17)000578R1_SOL005:_Refactor_PNFD_management - NFVSOL(17)000579_SOL005:_Annex_B.5_-_Add_NS_fault_management_mapping_operations_to_protocol_elements - NFVSOL(17)000604r1_SOL005:_NS_LCM_interface_-_edit_note_fix - NFVSOL(17)000607r1_SOL005:_NSD_management_interface_-_consistency_fix - NFVSOL(17)000608r1_SOL005:_NSD_management_interface_-_partial_download - NFVSOL(17)000610r1_SOL005:_NS_LCM_interface_-_edit_note_fix2 - NFVSOL(17)000613_SOL005:_VNF_package_management_interface_delete_VNF_package_flow - NFVSOL(17)000623R1_SOL005:_Resolve_editor's_note_in_clause 6.5.3.33_(HealNsData) - NFVSOL(17)000625_SOL005:_Remove_"pnfdName"-related_editor's_note_in_clause_5.5.2.5 - NFVSOL(17)000627r1_S0L005:_Use_of_verbal_forms_for_the_expression_of_provisions - NFVSOL(17)000628R1_SOL005:_Miscellaneous technical improvements - NFVSOL(17)000640_SOL005:_Update_sequence_diagram_for_the_get_operations_status_operations
November 2017	0.9.1	Clean-up done by ***editHelp!*** E-mail: mailto:edithelp@etsi.org
November 2017	0.10.0	Version 0.10.0 based on contributions that were agreed at the NFVSOL#42 and NFVSOL#43 meetings and during Email Approval (EA) resulting from the NFVSOL#41 meeting. - NFVSOL(17)000612R2_SOL005:_VNF_package_management_interface_refactoring_operations_and_data_types - NFVSOL(17)000632R2_SOL005_-_Editorial_changes - NFVSOL(17)000633R3_SOL005:_Sequence_diagram_for_the_deletion_of_an_individual_PNF_descriptor_reource - NFVSOL(17)000639R2_SOL005:_Sequence_diagram_for_the_deletion_of_an_individual_NS_descriptor_resource - NFVSOL(17)000645R1_SOL005:_Adding 405 response - NFVSOL(17)000648_SOL005:_Addition_of_the_notes_for_identifier_in_the_resource_URI - NFVSOL(17)000657_SOL005:_Flow_of_the_creation_of_an_individual_NS_descriptor_resource - NFVSOL(17)000658R1_SOL005:_Flow_of_the_creation_of_an_individual_PNF_descriptor_resource - NFVSOL(17)000659_SOL005:_Resolution_of_Rapporteur's_Notes_in_Annex_A - NFVSOL(17)000660_SOL005_Flow_of_the_querying_reading_of_NS_descriptor_resources - NFVSOL(17)000662_SOL005_Flow_of_the_querying_reading_of_PNF_descriptor_resources - NFVSOL(17)000664_SOL005_Proposed_resolution_of_clause_4_editor's note

Date	Version	Information about changes
		- NFVSOL(17)000665_SOL005_Proposed_resolution_of_clause_4.2_editor's note - NFVSOL(17)000678R1_SOL005_Update_NFP_related_data_types - NFVSOL(17)000679_SOL005_Add_VNF_package_state_model_to_annex_C - NFVSOL(17)000682_SOL005_Flow_of_the_update_of_an_individual_NS_descriptor_resource Version 0.10.0 also reflects additional clean-up done by ***editHelp!*** - E-mail: mailto:edithelp@etsi.org
December 2017	0.11.0	Version 0.11.0 based on contributions that were agreed at the NFVSOL#45 meeting. - NFVSOL(17)000606R3_SOL005:_Update_data_type_NsLcmOpOcc_and_NsLcmOperationOccurrenceNotification - NFVSOL(17)000609R3_SOL005:_NS LCM interface_-_Error handling enhancement - NFVSOL(17)000614R3_SOL005:_VNF_package_management_interface_upload_VNF_content_through_external_link - NFVSOL(17)000649R2:_SOL005:_6.5.3.2_OperateVnfData - NFVSOL(17)000672R2_SOL005:_Resolution_of_editor's_note_on_structure_of_theNSD_zip_file_in_clause_5.4.4.3.2 - NFVSOL(17)000680R2:_SOL005:_Update_ResourceHandle_datatype - NFVSOL(17)000683R2_SOL005:_Remove_pnfdInfoId_and_related_editor's_note_from_clause_5.5.2.6 - NFVSOL(17)000701R1_SOL005:_NS_LCM_interface_-_Sequence_diagram_for_continue_operation - NFVSOL(17)000702R1_SOL005:_Update_the_NSD_state_diagram_in _annex - NFVSOL(17)000703R1_SOL005:_NS_LCL_interface_-_resolve_the_editor's_note_about_resource_changes_in_NsLcmOperationOccurrenceNotification - NFVSOL(17)000704R1_SOL005:_Resolve_the_editor's_notes_on_NS_PM_interface - NFVSOL(17)000705_SOL005:_Resolve_the_editor's_notes_on_ExtVirtualLinkData_and_ExtManagedVirtualLinkData - NFVSOL(17)000707R2_SOL005:_Sequence_diagram_for_the_uploading_of_NSD_content - NFVSOL(17)000708R1_SOL005:_Sequence_diagram_for_the_uploading_of_PNFD_content - NFVSOL(17)000709R1_SOL005:_Resolution_of_editor's_note_on_the_NsLcmOpOcc_data_type_in_clause_6.4.9.3.2 - NFVSOL(17)000710_SOL005:_Resolution_of_editor's_note_in_clause 5.4.4.3.3_on_partial/chunking/resumable_upload - NFVSOL(17)000712R1:_SOL005_-_Additional_fields and values_for_NfpRules - NFVSOL(17)000713_SOL005_4.2_Consistency_of_URI_and OAuth - NFVSOL(17)000714R1:_SOL005:_Resolve_the_editor's_notes_on_ExtLinkPort and NsLinkPort - NFVSOL(17)000716R1_SOL005:_Double_subscriptions_for_notifications - NFVSOL(17)000720_SOL005:_Resolution_of_two_editor's_notes_on_pnfdInvariantId - NFVSOL(17)000721R1_SOL005:_Sequences of responses_and_notifications - NFVSOL(17)000727_SOL005:_Flow_of_the_fetching_of the_content_of_a_NSD - NFVSOL(17)000728_SOL005:_Flow_of_the_fetching_of the_content_of_a_PNFD - NFVSOL(17)000729_SOL005:_Update_to_the_flow_of_the_creation_of_the_individual_NS_descriptor_resource - NFVSOL(17)000731R1_SOL005:_Fix_description_of_unsupported_method_for_notification_endpoint - NFVSOL(17)000732R1_SOL005:_Resolve_editor's_note_on_checksum - NFVSOL(17)000738_SOL005_Add_Update PNFD Info_operation_to_the_list_of_NSD_management_interface_operations_in_clause 5.1 - NFVSOL(17)000741_SOL005:_Fix_the_inconsisistency_related_to_NS_LCM_operation_state - NFVSOL(17)000742_SOL005:_Add_PNFD_Notifications - NFVSOL(17)000743R1_SOL005:_Adding_normative_category_to_resource_and_methods_Tables - NFVSOL(17)000744_SOL005:_Align_PkgmNotificationsFilter_with_VnfInstanceSubscriptionFilter - NFVSOL(17)000748R1_SOL005:_Add_NS_change_notification_on_NS_LCM_interface

Date	Version	Information about changes
		- NFVSOL(17)000750:_SOL005:_Add_error_code_for_fetching_package_content,_vnfd_and_artifact_operations - NFVSOL(17)000755:_SOL005:_Remove_redundant_description_of_vnfConfigurableProperties - NFVSOL(17)000759R2_SOL005_Add ChangeVnfFlavourData - NFVSOL(17)000760_SOL005_6.5.3.2_Add_changedInfo_to AffectedVnf_data_type - NFVSOL(17)000762R1_SOL005:_Authorization of API requests_and_notifications
December 2017	0.12.0	Version 0.12.0 based on contributions that were agreed at the NFVSOL#46 meeting and during Email Approval (EA) following the NFVSOL#45 meeting. - NFVSOL(17)000603R1_SOL005:_NS_LCM_interface_-_network_address - NFVSOL(17)000745_SOL005: Change_the_name_of_the "ScaleInfo" data type to "VnfScaleInfo" - NFVSOL(17)000747R2_SOL005:_Add_and_modify_PNF_on_NS_LCM_interface - NFVSOL(17)000752_SOL005:_Change_"pnfdInfoStateType"_to_"pnfdOnboardingStateType"_ to resolve_editor's_note_in_clause_5.5.4.6 - NFVSOL(17)000761_SOL005:_Resolve_NsInstanceSubscriptionFilter_editor's_note_(Issue Gen.3) - NFVSOL(17)000763_SOL005:_Align_normative_statements_in_trigger_conditions_mirror_734 - NFVSOL(17)000767_SOL005:_Authorization_method_negotiation - NFVSOL(17)000769_SOL005:_Annex_A.6_operation_name_and_resource_URI alignment - NFVSOL(17)000770_SOL005:Add/remove_notes_about_race_conditions - NFVSOL(17)000772_SOL005:_Add_get_method_support_on_the_notification_endpoint_resource_in_the_NSD_and_NS_LCM interfaces - NFVSOL(17)000776_SOL005:_Alignment_of_timeStamp_attribute_name - NFVSOL(17)000777_SOL005:_Change "NsLinkPort" to "NsLinkPortInfo" - NFVSOL(17)000779_SOL005:_Precondition_for_VNF_package_deletion - NFVSOL(17)000784_SOL005:_Add_a_pointer_to_annex_B_for_the_NSD_state_model_in_NsdInfo
February 2018	2.4.1	Publication
March 2018	2.4.2	Version 2.4.2 based on contributions that were agreed at the NFVSOL#55 meeting. - NFVSOL(18)000047_SOL005ed251_API_authorization_clarification - NFVSOL(18)000084_SOL005ed251_Making_authorixation_negotiation_more_flexible
April 2018	2.4.3	Version 2.4.3 based on contributions that were agreed at the NFVSOL#59, NFVSOL#60, and NFVSOL#61 meetings (including EA) - NFVSOL(18)000098_SOL005ed251_Empty_collections_clarification_addressing_Plugtest^{TM} issue - NFVSOL(18)000137_SOL005ed251_Disambiguating artifactPath - NFVSOL(18)000155R1_SOL005ed251_Fix cardinality of the operationParams attribute in the NsLcmOpOcc data type
May 2018	2.4.4	Version 2.4.4 based on contributions that were agreed at the NFVSOL#62, NFVSOL#64, and NFVSOL#65 meetings (including EA) - NFVSOL(18)000167_SOL005ed251:_fixing_tracker_issue_007748 - NFVSOL(18)000177_SOL005ed251:_Correct_description_of_POST_method_on_NS_descriptors_resourc - NFVSOL(18)000184_SOL005ed251:_Change_the_cardinality_of_the_subscriptionId_atttribute_in_the_NSD_management,_NS_LCM,_and_VNF_package_management_notifications - NFVSOL(18)000233_SOL005ed251:_Define_userDefinedData_attribute_consistently - NFVSOL(18)000235_SOL005ed251:_Remove_reference_to_the_note_in_the_description_of_the_pnfdInvariantId_attribute_in_clause_5.5.2.5 - NFVSOL(18)000238_SOL005ed251: Updating_ JSON_RFC_reference - NFVSOL(18)000243_SOL005ed251:_VnfPkgm_small_fix - NFVSOL(18)000248_SOL005ed251:_Version_Management - NFVSOL(18)000251R1_SOL005ed251: Move_ResourceHandle_to common_data_types_in_clause_4.4.1.6

Date	Version	Information about changes
July 2018	2.4.5	Version 2.4.5 based on contributions that were agreed at the NFVSOL#66 and NFVSOL#67 (including EA) - NFVSOL(18)000244_SOL005ed251:_Different_names_for_virtual_link_descriptor_id - NFVSOL(18)000245_SOL005ed251:_Adding_status_codes - NFVSOL(18)000259R1_SOL005ed251:_Add_two … monitoringParameter_data_types - NFVSOL(18)000279_SOL005ed251:_Attribute_selectors - NFVSOL(18)000280_SOL005ed251:_Fixing_the_sequence_of_400_response_code_definitions - NFVSOL(18)000281_SOL005ed251:_Small_fix_to_the_description_of_the_400_error_code - NFVSOL(18)000311_SOL005ed251:_Small_fix_replace_queried_by_read - NFVSOL(18)000318_SOL005ed251:_Add_IFA027_reference - NFVSOL(18)000319_SOL005ed251:_Attribute_filters - NFVSOL(18)000320_SOL005ed251: String_and_number_data_types - NFVSOL(18)000321_SOL005ed251:_Mirror_of_NFVSOL(18)000341r2 - NFVSOL(18)000332R2_SOL005ed251:_Add_annex_with_a_reference_to_OpenAPI_repository - NFVSOL(18)8)000334_SOL005ed251:_Add_VL_profile_id_in_NsrtualLinkInfo_data_type - NFVSOL(18)000341_SOL005ed251: Clarifying_association_from VnfLinkPort to VnfcCp and VnfExtCp
July 2018	2.4.6	Version 2.4.6 based on contributions that were agreed at the NFVSOL#68 and NFVSOL#69 (including EA) - NFVSOL(18)000347_SOL005ed251:_Normative_attribute_filters_support - NFVSOL(18)000350_SOL005ed251:_Metadata_for CPs_and_extCPs - NFVSOL(18)000355_SOL005ed251:_Add_relative_URIs_for_links_in_notifications - NFVSOL(18)000356_SOL005ed251:_Retry_as_reaction_to_error_responses_during_notification_delivery - NFVSOL(18)000357_SOL005ed251:_NestedNsInstanceData_for_the_InstantiateNs_and_UpdateNs_operations - NFVSOL(18)000358_SOL005ed251:_Add_ParamsForNestedNS_to_the_InstantiateNs_operations - NFVSOL(18)000359_SOL005ed251:_Clarify_linkage_between_vnfcCP_and vnfcExtCP - NFVSOL(18)000360_SOL005ed251:_Attach metadata to extCPs - NFVSOL(18)000391_SOL005ed251:_Define_minor_version_number - NFVSOL(18)000392_SOL005ed251:_Attribute_selector_attribute_filter_small_fixes
July 2018	2.4.7	Version 2.4.7 based on contributions that were agreed at NFVSOL#71 (including EA) - NFVSOL(18)000361_SOL005ed251:_Fix_NFP_Management - NFVSOL(18)000438_SOL005ed251:_Ensure_consistency_with_SOL003_on_VnfLinkPortInfo_and_VnfExtCpInfo
August 2018	2.4.8	Version 2.4.8 based on contributions that were agreed at NFVSOL#72 and NFVSOL#72a (including EA) - NFVSOL(18)000456R1_SOL005ed251:_Version_management - NFVSOL(18)000458_SOL005ed251:_Version signaling - NFVSOL(18)000461R1_SOL005ed251: Define_patch_version_number - NFVSOL(18)000462_SOL005ed251: Closing pagination gap - NFVSOL(18)000473R1_SOL005ed251: Add_note_to_MAJOR_version_field

History

Document history		
V2.4.1	February 2018	Publication
V2.5.1	September 2018	Publication