



Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs

Document Preview

[ETSI GS NFV-SOL 013 V5.2.1 \(2024-12\)](https://standards.iteh.ai/catalog/standards/etsi/8c789494-d519-446d-afa7-e751adf8ac46/etsi-gs-nfv-sol-013-v5-2-1-2024-12)

<https://standards.iteh.ai/catalog/standards/etsi/8c789494-d519-446d-afa7-e751adf8ac46/etsi-gs-nfv-sol-013-v5-2-1-2024-12>

Disclaimer

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

Reference

RGS/NFV-SOL013ed521

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

Important notice

The present document can be downloaded from the
[ETSI Search & Browse Standards](#) application.

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

Users should be aware that the present document may be revised or have its status changed,
this information is available in the [Milestones listing](#).

If you find errors in the present document, please send your comments to
the relevant service listed under [Committee Support Staff](#).

If you find a security vulnerability in the present document, please report it through our
[Coordinated Vulnerability Disclosure \(CVD\)](#) program.

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2024.
All rights reserved.

Contents

Intellectual Property Rights	5
Foreword.....	5
Modal verbs terminology.....	5
1 Scope	6
2 References	6
2.1 Normative references	6
2.2 Informative references.....	7
3 Definition of terms, symbols and abbreviations.....	7
3.1 Terms.....	7
3.2 Symbols.....	7
3.3 Abbreviations	7
4 HTTP usage.....	8
4.1 URI structure and supported content formats.....	8
4.2 Usage of HTTP header fields	9
4.2.1 Introduction.....	9
4.2.2 Request header fields	9
4.2.3 Response header fields.....	10
5 Result set control.....	11
5.1 Introduction	11
5.2 Attribute-based filtering	11
5.2.1 Overview and example (informative)	11
5.2.2 Specification	12
5.3 Attribute selectors.....	14
5.3.1 Overview and example (informative)	14
5.3.2 Specification	14
5.3.2.1 GET request	14
5.3.2.2 GET response	15
5.4 Handling of large query results	15
5.4.1 Overview	15
5.4.2 Specification	16
5.4.2.1 Alternatives	16
5.4.2.2 Error response	16
5.4.2.3 Paged response	16
6 Error reporting.....	17
6.1 Introduction	17
6.2 General mechanism	17
6.3 Type: ProblemDetails.....	17
6.4 Common error situations	17
7 Common data types	20
7.1 Structured data types	20
7.1.1 Introduction.....	20
7.1.2 Type: Object	20
7.1.3 Type: Link	20
7.1.4 Type: NotificationLink	20
7.1.5 Type: KeyValuePairs.....	20
7.1.6 Type: ApiVersionInformation	20
7.1.7 Type: Checksum	21
7.2 Simple data types and enumerations	21
7.2.1 Introduction.....	21
7.2.2 Simple data types.....	21
7.2.3 Enumerations	22
8 Authorization of API requests and notifications	22

8.1	Introduction	22
8.2	Flows (informative)	23
8.2.1	General	23
8.2.2	Authorization of API requests using OAuth 2.0 access tokens	23
8.2.3	Authorization of API requests using TLS certificates	25
8.2.4	Authorization of notifications using the HTTP Basic authentication scheme	25
8.2.5	Authorization of notifications using OAuth 2.0 access tokens	25
8.2.6	Authorization of notifications using TLS certificates	28
8.2.7	Authorization of API requests using OAuth 2.0 certificate-bound access tokens	28
8.3	Specification	30
8.3.1	Introduction	30
8.3.2	General mechanism	30
8.3.3	Authorizing API requests	31
8.3.4	Authorizing the sending of notifications	31
8.3.5	Client roles	33
8.3.6	Support of authorization	34
8.3.6.1	Authorization of API requests	34
8.3.6.2	Authorization of notification requests	34
8.3.7	Authorization scope values	34
9	Version management	35
9.1	Version identifiers and parameters	35
9.1.1	Version identifiers	35
9.1.2	Version parameters	35
9.2	Rules for incrementing version identifier fields	36
9.2.1	General	36
9.2.2	Examples of backward and non-backward compatible changes	36
9.3	Version information retrieval	37
9.3.1	General	37
9.3.2	Resource structure and methods	37
9.3.3	Resource: API versions	38
9.3.3.1	Description	38
9.3.3.2	Resource definition	38
9.3.3.3	Resource methods	38
9.3.3.3.1	POST	38
9.3.3.3.2	GET	39
9.3.3.3.3	PUT	39
9.3.3.3.4	PATCH	39
9.3.3.3.5	DELETE	39
9.4	Version signalling	39
Annex A (informative): Change history		41
History		43

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

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

Modal verbs terminology

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

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

1 Scope

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

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at [ETSI docbox](#).

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] Void.
- [2] [IETF RFC 3339](#): "Date and Time on the Internet: Timestamps".
- [3] [IETF RFC 3986](#): "Uniform Resource Identifier (URI): Generic Syntax".
- [4] Void.
- [5] [IETF RFC 5246](#): "The Transport Layer Security (TLS) Protocol Version 1.2".
- [6] [IETF RFC 6585](#): "Additional HTTP Status Codes".
- [7] [IETF RFC 6749](#): "The OAuth 2.0 Authorization Framework".
- [8] [IETF RFC 6750](#): "The OAuth 2.0 Authorization Framework: Bearer Token Usage".
- [9] [IETF RFC 8259](#): "The JavaScript Object Notation (JSON) Data Interchange Format".
- [10] Void.
- [11] Void.
- [12] Void.
- [13] Void.
- [14] [IETF RFC 7617](#): "The 'Basic' HTTP Authentication Scheme".
- [15] [IETF RFC 7807](#): "Problem Details for HTTP APIs".
- [16] [IETF RFC 6901](#): "JavaScript Object Notation (JSON) Pointer".
- [17] [IETF RFC 8288](#): "Web Linking".
- [18] [Semantic Versioning 2.0.0](#).
- [19] [IETF RFC 4229](#): "HTTP Header Field Registrations".
- [20] [ETSI GS NFV-SEC 022](#): "Network Functions Virtualisation (NFV) Release 4; Security; Access Token Specification for API Access".

- [21] [ETSI TS 133 210](#): "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Network Domain Security (NDS); IP network layer security (3GPP TS 33.210)".
- [22] [IETF RFC 8446](#): "The Transport Layer Security (TLS) Protocol Version 1.3".
- [23] [IETF RFC 7515](#): "JSON Web Signature (JWS)".
- [24] [IETF RFC 9110](#): "HTTP Semantics".
- [25] [IETF RFC 8705](#): "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] ETSI GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".
- [i.2] Void.
- [i.3] [IANA](#): "Hypertext Transfer Protocol (HTTP) Status Code Registry".
- [i.4] [ETSI NFV OpenAPI repository](#).
- [i.5] [JSON Schema Validation](#): "A Vocabulary for Structural Validation of JSON".
- [i.6] [OpenAPI™ Specification](#).
- [i.7] [IANA](#): "Hash Function Textual Names".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
EM	Element Manager
ETSI	European Telecommunications Standards Institute
GMT	Greenwich Mean Time
GS	Group Specification

HATEOAS	Hypermedia As The Engine Of Application State
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
MAC	Medium Access Control
MANO	MANagement and Orchestration
MIME	Multipurpose Internet Mail Extensions
NFV	Network Functions Virtualisation
NFVO	NFV Orchestrator
REST	REpresentational State Transfer
RFC	Request For Comments
TLS	Transport Layer Security
URI	Uniform Resource Identifier
VIM	Virtualised Infrastructure Manager
VNF	Virtualised Network Function
VNFM	VNF Manager

4 HTTP usage

4.1 URI structure and supported content formats

This clause specifies the URI prefix and the supported formats applicable to the RESTful NFV-MANO APIs.

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

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

where:

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

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

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

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

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

HTTP shall be run as an application protocol over TLS, a combination that is known as HTTPS. All APIs shall use TLS version 1.2 as defined by IETF RFC 5246 [5] or later. Versions of TLS earlier than 1.2 shall neither be supported nor used.

NOTE 1: The HTTP protocol elements mentioned in the RESTful NFV-MANO API specifications originate from the HTTP specification; HTTPS runs the HTTP protocol on top of a TLS layer. The RESTful NFV-MANO specifications therefore use the statement above to mention "HTTP request", "HTTP header", etc., without explicitly calling out whether or not these are run over TLS.

TLS implementations shall meet or exceed the security algorithm, key length and strength requirements specified in clause 6.2.3 (if TLS version 1.2 as defined by IETF RFC 5246 [5] is used) or clause 6.2.2 (if TLS version 1.3 as defined by IETF RFC 8446 [22] is used) of ETSI TS 133 210 [21] (3GPP Release 16 or later).

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

NOTE 2: This means that characters not part of this allowed set are escaped using percent-encoding as defined by IETF RFC 3986 [3].

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

NOTE 3: A request URI parameter is denoted by a string in curly brackets, e.g. {subscriptionId}.

4.2 Usage of HTTP header fields

4.2.1 Introduction

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

4.2.2 Request header fields

This clause describes the usage of selected HTTP header fields of the request messages in the RESTful NFV-MANO APIs. The HTTP header fields used in the request messages are specified in table 4.2.2-1.

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

Header field name	Reference	Example	Descriptions
Accept	IETF RFC 9110 [24]	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 9110 [24]	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 9110 [24]	Bearer mF_9.B5f-4.1JqM	The authorization token for the request. Details are specified in clause 8.3.
Range	IETF RFC 9110 [24]	1 000-2 000	Requested range of bytes from a file.
Version	IETF RFC 4229 [19]	1.2.0 or 1.2.0-impl:example.com:myProduct:4	Version of the API requested to use when responding to this request.
If-Unmodified-Since	IETF RFC 9110 [24]	Sat, 29 Oct 1994 19:43:31 GMT	Used to make the request method conditional on the selected resource representation's last modification date being earlier than or equal to the date provided in the field-value. If the condition is not met, the request fails with a "412 Precondition Failed" response.

Header field name	Reference	Example	Descriptions
If-Match	IETF RFC 9110 [24]	"xyzzzy" or "xyzzzy", "r2d2xxxx", "c3piozzzz" or *	Used to make the request method conditional on the recipient origin server either having at least one current representation of the target resource, when the field-value is "*", or having a current representation of the target resource that has an entity-tag matching a member of the list of entity-tags provided in the field-value. If the condition is not met, the request fails with a "412 Precondition Failed" response.

4.2.3 Response header fields

This clause describes the usage of selected HTTP header fields of the response messages in the RESTful NFV-MANO APIs. The HTTP header fields used in the response messages are specified in table 4.2.3-1.

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

Header field name	Reference	Example	Descriptions
Content-Type	IETF RFC 9110 [24]	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 9110 [24]	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 RESTful NFV-MANO APIs this header field is also used if the response status code is 202 and a new resource was created.
WWW-Authenticate	IETF RFC 9110 [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 9110 [24]	bytes	Used by the server to signal whether or not it supports ranges for certain resources.
Content-Range	IETF RFC 9110 [24]	bytes 21 010 - 47 021/47 022	Signals the byte range that is contained in the response, and the total length of the file.
Retry-After	IETF RFC 9110 [24]	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.
Link	IETF RFC 8288 [17]	<http://example.com/resources?nextpage_opaque_marker=abc123>; rel="next"	Reference to other resources. Used for paging in the present document, see clause 5.4.2.1.
Version	IETF RFC 4229 [19]	1.2.0 or 1.2.0-impl:example.com:myProduct:4	Version of the API requested to use when responding to this request.

Header field name	Reference	Example	Descriptions
ETag	IETF RFC 9110 [24]	"33a64df551425fcc55e4d42a148795d9f25f89d4" or W/"0815"	Used to provide the current entity-tag for the selected resource representation. It can be sent in "200 OK", "201 Created" and "204 No Content" responses.
Last-Modified	IETF RFC 9110 [24]	Tue, 15 Nov 1994 12:45:26 GMT	Used to provide a timestamp indicating the date and time at which the server believes the selected resource representation was last modified. It can be sent in "200 OK", "201 Created" and "204 No Content" responses.

5 Result set control

5.1 Introduction

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

5.2 Attribute-based filtering

5.2.1 Overview and example (informative)

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

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

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

EXAMPLE 1: Objects:

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

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

EXAMPLE 2: Unfiltered GET:

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

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

EXAMPLE 3: GET with filter:

```
Request:
GET .../container?filter=(eq,weight,100)
Response:
[
```

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

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

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

5.2.2 Specification

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

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

where:

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

"AttrName" is the name of one attribute in the data type that defines the representation of the resource. The slash ("/") character in "simpleFilterExprOne" and "simpleFilterExprMulti" allows concatenation of <attrName> entries to filter by attributes deeper in the hierarchy of a structured document. The special attribute name "@key" refers to the key of a map.

EXAMPLE: Referencing the key of a map:

```
GET .../resource?filter=(eq,mymap/@key,abc123)
```

The elements "opOne" and "opMulti" stand for the comparison operators (accepting one comparison value or a list of such values). If the expression has concatenated <attrName> entries, it means that the operator is applied to the attribute addressed by the last <attrName> entry included in the concatenation. All simple filter expressions are combined by the "AND" logical operator, denoted by ";".

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

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

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