**3GPP TSG-SA5 Meeting #142-e *S5-222095***

**e-meeting, 4 - 12 April 2022**

|  |
| --- |
| *CR-Form-v12.1* |
| **CHANGE REQUEST** |
|  |
|  | **32.158** | **CR** | **DraftCR** | **rev** | **-** | **Current version:** | **16.5.0** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network | **X** | Core Network | **X** |

|  |
| --- |
|  |
| ***Title:***  | Rel-16 CR 32.158 Correct examples in Annex A |
|  |  |
| ***Source to WG:*** | Nokia, Nokia Shanghai Bell |
| ***Source to TSG:*** | S5 |
|  |  |
| ***Work item code:*** | TEI16, REST\_SS |  | ***Date:*** | 2022-03-25 |
|  |  |  |  |  |
| ***Category:*** | **F** |  | ***Release:*** | Rel-17 |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-8 (Release 8)Rel-9 (Release 9)Rel-10 (Release 10)Rel-11 (Release 11)…Rel-15 (Release 15)Rel-16 (Release 16)Rel-17 (Release 17)Rel-18 (Release 18)* |
|  |  |
| ***Reason for change:*** | The REST guidelines are slightly underspecified as to a few aspects and require some clarifications to produce fully disambiguous definitions. These clarifications are provided in other CRs. This CR mirrors these changes into the examples in Annex A. |
|  |  |
| ***Summary of change:*** | The examples are clarified and disambiguated. |
|  |  |
| ***Consequences if not approved:*** | Unclear examples impair interoperability. |
|  |  |
| ***Clauses affected:*** | A.1, A.2.1, A.2.2, A.2.3, A.3.1, A.3.2, A.3.3, A.3.4 (new), A.4.1, A.4.2, A.4.3, A.4.4 (new), A.5, A.6.1, A.6.2, A.6.3, A.6.4, A.7.1, A.7.2,  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  | **X** |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  | **X** |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  | **X** |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** |  |
|  |  |
| ***This CR's revision history:*** |  |

|  |
| --- |
| **First modification** |

Annex A (informative):
Examples

# A.1 Example data model

The following JSON instance document is used for the examples in this clause.

|  |
| --- |
| { "SubNetwork": [ { "id": "SN1", "objectClass": "SubNetwork", "objectInstance": "SubNetwork=SN1", "attributes": { "userLabel": "Berlin NW", "userDefinedNetworkType": "5G", "plmnId-id": { "mcc": 456, "mnc": 789 } }, "ManagedElement": [ { "id": "ME1", "objectClass": "ManagedElement", "objectInstance": "SubNetwork=SN1,ManagedElement=ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" }, "XyzFunction": [ { "id": "XYZF1", "objectClass": "XyzFunction", "objectInstance":"SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } }, { "id": "XYZF2", "objectClass": "XyzFunction", "objectInstance":"SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF2", "attributes": { "attrA": "abc", "attrB": 552 } } ] }, { "id": "ME2", "objectClass": "ManagedElement", "objectInstance": "SubNetwork=SN1,ManagedElement=ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } } ], "PerfMetricJob": [ { "id": "PMJ1", "objectClass": "PerfMetricJob", "objectInstance": "SubNetwork=SN1,PerfMetricJob=PMJ1", "attributes": { "granularityPeriod": "5", "perfMetrics": [ "Metric1", "Metric2" ], "objectInstances": [ "Obj1", "Obj2" ] } } ], "ThresholdMonitor": [ { "id": "TM1", "objectClass": "ThresholdMonitor", "objectInstance": "SubNetwork=SN1,ThresholdMonitor=TM1", "attributes": { "metric": "Metric1", "thresholdLevels": [ { "level": "1", "thresholdValue": 10 }, { "level": "2", "thresholdValue": 20 }, { "level": "3", "thresholdValue": 30 } ] } } ] } ]} |

The corresponding JSON schema is

|  |
| --- |
| { "SubNetwork": { "type": "array", "items": { "type": "object", "properties": { "id": { "type": "string" }, "objectClass": { "type": "string" }, "objectInstance": { "type": "string" }, "attributes": { "type": "object", "properties": { "userLabel": { "type": "string" }, "userDefinedNetworkType": { "type": "string" }, "plmnId": { "type": "object", "properties": { "mcc": { "type": "integer" }, "mnc": { "type": "integer" } } } } }, "ManagedElement": { "type": "array", "items": { "type": "object", "properties": { "id": { "type": "string" }, "objectClass": { "type": "string" }, "objectInstance": { "type": "string" }, "attributes": { "type": "object", "properties": { "userLabel": { "type": "string" }, "vendorName": { "type": "string" }, "location": { "type": "string" } } }, "XyzFunction": { "type": "array", "items": { "type": "object", "properties": { "id": { "type": "string" }, "objectClass": { "type": "string" }, "objectInstance": { "type": "string" }, "attributes": { "type": "object", "properties": { "attributeA": { "type": "string" }, "attributeB": { "type": "integer" } } }, "required": ["id"] } } }, "required": ["id"] } } }, "PerfMetricJob": { "type": "array", "items": { "type": "object", "properties": { "id": { "type": "string" }, "objectClass": { "type": "string" }, "objectInstance": { "type": "string" }, "attributes": { "type": "object", "properties": { "granularityPeriod": { "type": "integerstring" }, "perfMetrics": { "type": "array", "items": { "type": "string" } } }, "objectInstances": { "type": "array", "items": { "type": "string" } } }, "required": ["id"] } } }, "ThresholdMonitor": { "type": "array", "items": { "type": "object", "properties": { "id": { "type": "string" }, "objectClass": { "type": "string" }, "objectInstance": { "type": "string" }, "attributes": { "type": "object", "properties": { "thresholdLevels": { "type": "array", "items": { "type": "object", "properties": { "level": { "type": "string" }, "thresholdValue": { "type": "integer" } } } } } }, "required": ["id"] } } }, "required": ["id"] } } }} |

The corresponding XML instance document is provided below as well. It can be helpful when evaluating XPath expressions.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><SubNetwork> <id>SN1</id> <objectClass>SubNetwork</objectClass> <objectInstance>SubNetwork=SN1</objectInstance> <attributes> <userLabel>Berlin NW</userLabel> <userDefinedNetworkType>5G</userDefinedNetworkType> <plmnId> <mcc>456</mcc> <mnc>789</mnc> </plmnId> </attributes> <ManagedElement> <id>ME1</id> <objectClass>ManagedElement</objectClass> <objectInstance>SubNetwork=SN1,ManagedElement=ME1</objectInstance> <attributes> <userLabel>Berlin NW 1</userLabel> <vendorName>Company XY</vendorName> <location>TV Tower</location> </attributes> <XyzFunction> <id>XYZF1</id> <objectClass>XyzFunction</objectClass> <objectInstance>SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF1</objectInstance> <attributes> <attrA>xyz</attrA> <attrB>551</attrB> </attributes> </XyzFunction> <XyzFunction> <id>XYZF2</id> <objectClass>XyzFunction</objectClass> <objectInstance>SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF2</objectInstance> <attributes> <attrA>abc</attrA> <attrB>552</attrB> </attributes> </XyzFunction> </ManagedElement> <ManagedElement> <id>ME2</id> <objectClass>ManagedElement</objectClass> <objectInstance>SubNetwork=SN1,ManagedElement=ME2</objectInstance> <attributes> <userLabel>Berlin NW 2</userLabel> <vendorName>Company XY</vendorName> <location>Grunewald</location> </attributes> </ManagedElement> <PerfMetricJob> <id>PMJ1</id> <objectClass>PerfMetricJob</objectClass> <objectInstance>SubNetwork=SN1,PerfMetricJob=PMJ1</objectInstance> <attributes> <granularityPeriod>5</granularityPeriod> <perfMetrics>Metric1</perfMetrics> <perfMetrics>Metric2</perfMetrics> <objectInstances>Obj1</objectInstances> <objectInstances>Obj2</objectInstances> </attributes> </PerfMetricJob> <ThresholdMonitor> <id>TM1</id> <objectClass>ThresholdMonitor</objectClass> <objectInstance>SubNetwork=SN1,ThresholdMonitor=TM1</objectInstance> <attributes> <ThresholdLevels> <level>1</level> <thresholdValue>10</thresholdValue> </ThresholdLevels> <ThresholdLevels> <level>2</level> <thresholdValue>20<thresholdValue> </ThresholdLevels> <ThresholdLevels> <level>3</level> < thresholdValue>30</thresholdValue> </ThresholdLevels> </attributes> </ThresholdMonitor></SubNetwork> |

NOTE: The following examples do not always follow the URI structure specified in clause 4.4. For simplicity reasons, the path component "/{MnSName}/{MnSVersion}" is often omitted. Also the Domain Component (DC) is omitted in DNs carried by "objectInstance" attributes. Though this is a valid implementation as per 3GPP TS 32.300 [3], it is recommended to have "DC=example.org" or "DC=org, DC=example" as first components of DNs.

# A.2 Retrieval of resources

## A.2.1 Retrieval of a single complete resource with HTTP GET

To retrieve a complete "XyzFunction" resource the MnS Consumer might send the following request.

|  |
| --- |
| GET /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1 HTTP/1.1Host: example.orgAccept: application/json |

The response includes the resource representation

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 }} |

Alternatively, the response might include a key ("XyzFunction") specifying the class name of the returned resource

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "XyzFunction": [ { "id": "XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } } ]} |

In the example above "XyzFunction" is of type array to align with the JSON schema of "XyzFunction" defined in clause A.1. Alternatively, "XyzFunction" might also be an object, since the JSON schema specifying the response message body is not required to be identical to the JSON schema specifying the resources contained by a resource.

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "XyzFunction": { "id": "XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } }} |

Alternatively, when using a "data" object the response might look like

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "data": { "XyzFunction": { "id": "XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } } }} |

When building the response based on the flat response construction method, the response is given by

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json[ { "id": "XYZF1", "objectClass": "XyzFunction", "objectInstance": "SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } }] |

The exact syntax of the response body is specified by the JSON schema included in the concrete MnS definition. In current MnS definitions the format of the first example response above is used. This style is also followed in subsequent examples.

## A.2.2 Attribute and attribute field selection on a single resource

To retrieve only the "userLabel" attribute and the "mnc" attribute field of the "plmnId" attribute of the "SubNetwork", the MnS Consumer might send:

|  |
| --- |
| GET /SubNetwork=SN1?attributes=userLabel&fields=/attributes/plmnId/mcc HTTP/1.1Host: example.orgAccept: application/json |

Alternatively one might send as well

|  |
| --- |
| GET /SubNetwork=SN1?fields=/attributes/userLabel,/attributes/plmnId/mcc HTTP/1.1Host: example.orgAccept: application/json |

The response contains only the selected attribute "userLabel" and the selected attribute field "mnc":

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "SN1", "attributes": { "userLabel": "Berlin NW", "plmnId": { "mnc": 789 } }} |

In this example, the MnS Consumer retrieves the "userLabel" and "vendorName" of the "ManagedElement" whose "id" is equal to "ME1":

|  |
| --- |
| GET /SubNetwork=SN1/ManagedElement=ME1?attributes=userLabel,vendorName HTTP/1.1Host: example.orgAccept: application/json |

The MnS Producer responds as follows:

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY" }} |

The following request selects all attributes:

|  |
| --- |
| GET /SubNetwork=SN1/ManagedElement=ME1?fields=/attributes HTTP/1.1Host: example.orgAccept: application/json |

It is thus identical to:

|  |
| --- |
| GET /SubNetwork=SN1/ManagedElement=ME1 HTTP/1.1Host: example.orgAccept: application/json |

Both requests return the complete resource representation with all attributes:

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" }} |

The following request returns the first item of the "perfMetrics" attribute, which is of type array:

|  |
| --- |
| GET /SubNetwork=SN1/ManagedElement=ME1/PerfMetricJob=PMJ1?fields=attributes/perfMetrics/0 HTTP/1.1Host: example.orgAccept: application/json |

Note indices start with "0" in JSON Pointer. The response looks like:

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "PMJ1", "attributes": { "perfMetrics": [ "Metric1" ] }} |

## A.2.3 Retrieval of multiple complete resources using scoping and filtering

The following example selects the "SubNetwork" as base object at scope level "0" and all objects at scope level "1":

|  |
| --- |
| GET /SubNetwork=SN1?scopeType=BASE\_SUBTREE&scopeLevel=1 HTTP/1.1Host: example.orgAccept: application/json |

All objects at scope level "1" are included in the response irrespective of their object class. When using the hierarchical response construction method, the response looks as follows:

|  |
| --- |
| { "id": "SN1", "attributes": { "userLabel": "Berlin NW", "userDefinedNetworkType": "5G", "plmnId": { "mcc": 456, "mnc": 789 } }, "ManagedElement": [ { "id": "ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" } }, { "id": "ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } } ], "PerfMetricJob": [ { "id": "PMJ1", "attributes": { "granularityPeriod": 5, "perfMetrics": [ "Metric1", "Metric2" ], "objectInstances": [ "Obj1", "Obj2" ] } } ], "ThresholdMonitor": [ { "id": "TM1", "attributes": { "metric": "Metric1", "thresholdLevels": [ { "level": "1", "thresholdValue": 10 }, { "level": "2", "thresholdValue": 20 }, { "level": "3", "thresholdValue": 30 } ] } } ]} |

The response constructed with the flat response construction method looks like:

|  |
| --- |
| [ { "id": "SN1", "objectClass": "SubNetwork", "objectInstance": "SubNetwork=SN1", "attributes": { "userLabel": "Berlin NW", "userDefinedNetworkType": "5G", "plmnId": { "mcc": 456, "mnc": 789 } } }, { "id": "ME1", "objectClass": "ManagedElement", "objectInstance": "SubNetwork=SN1,ManagedElement=ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" } }, { "id": "ME2", "objectClass": "ManagedElement", "objectInstance": "SubNetwork=SN1,ManagedElement=ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } }, { "id": "PMJ1", "ojectClass": "PerfMetricJob", "objectInstance": "SubNetwork=SN1,PerfMetricJob=PMJ1", "attributes": { "granularityPeriod": "5", "perfMetrics": [ "Metric1", "Metric2" ], "objectInstances": [ "Obj1", "Obj2" ] } }, { "id": "TM1", "ojectClass": "ThresholdMonitor", "objectInstance": "SubNetwork=SN1,ThresholdMonitor=TM1", "attributes": { "metric": "Metric1", "thresholdLevels": [ { "level": "1", "thresholdValue": 10 }, { "level": "2", "thresholdValue": 20 }, { "level": "3", "thresholdValue": 30 } ] } }] |

When only objects at scope level "1" are requested to be returned, the request looks like:

|  |
| --- |
| GET /SubNetwork=SN1?scopeType=BASE\_NTH\_LEVEL&scopeLevel=1 HTTP/1.1Host: example.orgAccept: application/json |

The response does not include the attributes of "SubNetwork" anymore, only its "id" is included:

|  |
| --- |
| { "id": "SN1", "ManagedElement": [ { "id": "ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" } }, { "id": "ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } } ], "PerfMetricJob": [ { "id": "PMJ1", "attributes": { "granularityPeriod": 5, "perfMetrics": [ "Metric1", "Metric2" ], "objectInstances": [ "Obj1", "Obj2" ] } } ], "ThresholdMonitor": [ { "id": "TM1", "attributes": { "metric": "Metric1", "thresholdLevels": [ { "level": "1", "thresholdValue": 10 }, { "level": "2", "thresholdValue": 20 }, { "level": "3", "thresholdValue": 30 } ] } } ]} |

Similarly, for reading all objects on scope level "2", the MnS Consumer may send:

|  |
| --- |
| GET /SubNetwork=SN1?scopeType=BASE\_NTH\_LEVEL&scopeLevel=2 HTTP/1.1Host: example.orgAccept: application/json |

When using the hierarchical response construction method, the response includes the complete representations of the "XyzFunction" objects. The "SubNetwork" and "ManagedElement" are present with their "id" only; they provide the containment nodes for the "XyzFunction" objects.

|  |
| --- |
| { "id": "SN1", "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } }, { "id": "XYZF2", "attributes": { "attrA": "abc", "attrB": 552 } } ] } ]} |

The "PerfMetricJob" and "ThresholdMonitor" are not included altogether, not even with the "id" only. This is because these nodes do not represent necessary path components to the scoped objects on the second level.

When using the flat response construction method, the response includes only "XyzFunction" objects without containment nodes. The "objectInstance" of each returned object is present in this case, as required in clause 6.1.4.

|  |
| --- |
| [ { "id": "XYZF1", "objectClass": "XyzFunction", "objectInstance": "SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF1", "attributes": { "attrA": "xyz", "attrB": 551 } }, { "id": "XYZF2", "objectClass": "XyzFunction", "objectInstance": "SubNetwork=SN1,ManagedElement=ME1,XyzFunction=XYZF2", "attributes": { "attrA": "abc", "attrB": 552 } }] |

The following example selects all objects on scope level "1" that have a "location" attribute whose value is equal to "Grunewald":

|  |
| --- |
| GET /SubNetwork=SN1?\ scopeType=BASE\_NTH\_LEVEL&scopeLevel=1\ filter=/\*/\*[attributes[location="Grunewald"]] HTTP/1.1Host: example.orgAccept: application/json |

The response includes one "ManagedElement" object only:

|  |
| --- |
| { "id": "SN1", "ManagedElement": [ { "id": "ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } } ]} |

The input document to the XPath filter is a document whose root node is the object identified by the path component of the URI and that includes the object representations of the scoped objects. In this example the root node is the "SubNetwork" without the "attributes" node. The input document includes all scoped objects on the scope level "1". These are the two "ManagedElement" objects and the "PerfMetricJob" object:

|  |
| --- |
| { "SubNetwork": { "id": "SN1", "ManagedElement": [ { "id": "ME1", "attributes": { "userLabel": "Berlin NW 1", "vendorName": "Company XY", "location": "TV Tower" } }, { "id": "ME2", "attributes": { "userLabel": "Berlin NW 2", "vendorName": "Company XY", "location": "Grunewald" } } ], "PerfMetricJob":[ { "id": "PMJ1", "attributes": { "granularityPeriod": 5, "perfMetrics": ["Metric1", "Metric2"], "objectInstances": ["Obj1", "Obj2"] } } ], "ThresholdMonitor": [ { "id": "TM1", "attributes": { "metric": "Metric1", "thresholdLevels": [ { "level": "1", "thresholdValue": 10 }, { "level": "2", "thresholdValue": 20 }, { "level": "3", "thresholdValue": 30 } ] } } ] }} |

An implementation may be based on available XPath tools. In that case the JSON document may have to be converted to a XML document:

|  |
| --- |
| <SubNetwork> <id>SN1</id> <ManagedElement> <id>ME1</id> <attributes> <userLabel>Berlin NW 1</userLabel> <vendorName>Company XY</vendorName> <location>TV Tower</location> </attributes> </ManagedElement> <ManagedElement> <id>ME2</id> <attributes> <userLabel>Berlin NW 2</userLabel> <vendorName>Company XY</vendorName> <location>Grunewald</location> </attributes> </ManagedElement> <PerfMetricJob> <id>PMJ1</id> <attributes> <granularityPeriod>5</granularityPeriod> <perfMetrics>Metric1</perfMetrics> <perfMetrics>Metric2</perfMetrics> <objectInstances>Obj1</objectInstances> <objectInstances>Obj2</objectInstances> </attributes> </PerfMetricJob> <ThresholdMonitor> <id>TM1</id> <attributes> <ThresholdLevels> <level>1</level> <thresholdValue>10</thresholdValue> </ThresholdLevels> <ThresholdLevels> <level>2</level> <thresholdValue>20</thresholdValue> </ThresholdLevels> <ThresholdLevels> <level>3</level> <thresholdValue>30</thresholdValue> </ThresholdLevels> </attributes> </ThresholdMonitor></SubNetwork> |

In this example the complete "ManagedElement" object is the result of applying the XPath expression:

|  |
| --- |
| <ManagedElement> <id>ME2</id> <attributes> <userLabel>Berlin NW 2</userLabel> <vendorName>Company XY</vendorName> <location>Grunewald</location> </attributes></ManagedElement> |

XPath predicates allow to specify also ranges. The following example selects objects on scope level "2" that have an attribute with name "attrB" whose value is equal to or greater than 552 and less than 562.

|  |
| --- |
| GET /SubNetwork=SN1?\ scopeType=BASE\_NTH\_LEVEL&scopeLevel=2\ filter=/\*/\*/\*[attributes[attrB>=552 and attrB<562]] HTTP/1.1Host: example.orgAccept: application/json |

The response includes one "XyzFunction" object only:

|  |
| --- |
| { "id": "SN1", "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF2", "attributes": { "attrA": "abc", "attrB": 552 } } ] } ]} |

An identical response is returned when using the following requests:

|  |
| --- |
| GET /SubNetwork=SN1?\ scopeType=BASE\_ALL\ filter=//\*[attributes[attrB>=552 and attrB<562]] HTTP/1.1Host: example.orgAccept: application/json |

or

|  |
| --- |
| GET /SubNetwork=SN1?\ scopeType=BASE\_SUBTREE&scopeLevel=2\ filter=//\*[attributes[attrB>=552 and attrB<562]] HTTP/1.1Host: example.orgAccept: application/json |

or

|  |
| --- |
| GET /SubNetwork=SN1?\ scopeType=BASE\_ALL\ filter=//XyzFunction[attributes[attrB>=552 and attrB<562]] HTTP/1.1Host: example.orgAccept: application/json |

This example returns the containment tree only.

|  |
| --- |
| GET /SubNetwork=SN1?scopeType=BASE\_ALL&attributes= HTTP/1.1Host: example.orgAccept: application/json |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ "id": "SN1", "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF1" }, { "id": "XYZF2" } ] }, { "id": "ME2" } ], "PerfMetricJob": [ { "id": "PMJ1" } ], "ThresholdMonitor": [ { "id": "TM1" } ]} |

When the MnS Consumer does not know the root object of the containment tree and wants to retrieve the complete tree starting with the root, the target URI needs to identify the resource above the root object. According to clause 4.4.2 this resource is identified by the path segment "/{MnSName}/{MnSVersion}", for example "/ProvMnS/1700". In this example, the empty "attributes" parameter is defined to return only the name-containment hierarchy but no attributes.

|  |
| --- |
| GET /ProvMnS/1700?scopeType=BASE\_ALL&attributes= HTTP/1.1Host: example.orgAccept: application/json |

The response is illustrated below. Properties of the MnS may be returned as siblings of "SubNetwork", as indicated in the example below by the placeholder "…".

|  |
| --- |
| HTTP/1.1 200 OKDate: Tue, 06 Aug 2019 16:50:26 GMTContent-Type: application/json{ ..., "SubNetwork": [ { "id": "SN1", "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF1" }, { "id": "XYZF2" } ] }, { "id": "ME2" } ], "PerfMetricJob": [ { "id": "PMJ1" } ], "ThresholdMonitor": [ { "id": "TM1" } ] } ]} |

# A.3 Creation of resources

## A.3.1 Creation of a resource with HTTP PUT

In this example a new "XyzFunction" resource is created. The target URI specifies the location of the new resource. The object class name of the resource to be created is present in the request. The "id" of the new resource is "XYZF3" and created by the MnS Consumer. The "id" contained in the resource representation carried in the request message body and the "id" in the target URI are identical.

|  |
| --- |
| PUT /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF3 HTTP/1.1Host: example.orgContent-Type: application/json{ "id": "XYZF3", "objectClass": "XyzFunction", "attributes": { "attrA": "ghi", "attrB": 553 }} |

The response contains the location header with the URI of the created resource. The response body includes the complete representation of the new resource. This is necessary, since the MnS Producer may have modified or added attributes compared to those received in the creation request. The name of the object class may or may not be present in the response.

|  |
| --- |
| HTTP/1.1 201 CreatedDate: Tue, 06 Aug 2019 16:50:26 GMTLocation: http://example.org/SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF3Content-Type: application/json{ "id": "XYZF3", "attributes": { "attrA": "ghi", "attrB": 553 }} |

## A.3.2 Creation of a resource with HTTP POST

When creating a new resource with POST the target URI identifies the parent resource of the new resource to be created. The identifier of the new resource is created by the MnS Producer, hence the "id" is equal to "null" in the POST request. If the "id" carries a value, then the MnS Producer may consider that value as a non-binding recommendation by the MnS Consumer. The request message body includes the object class name of the resource to be created.

|  |
| --- |
| POST /SubNetwork=SN1/ManagedElement=ME1 HTTP/1.1Host: example.orgContent-Type: application/json{ "id": "null", "objectClass": "XyzFunction", "attributes": { "attrA": "ghi", "attrB": 553 }} |

<add new example>

For the response body the same provisions as for resource creation with HTTP PUT apply.

|  |
| --- |
| HTTP/1.1 201 CreatedDate: Tue, 06 Aug 2019 16:50:26 GMTLocation: http://example.org/SubNetwork=SN1/ManagedElement=ME1/XyzFunction=123e4567-e89bContent-Type: application/json{ "id": "123e4567-e89b", "attributes": { "attrA": "ghi", "attrB": 553 }} |

When creating a root resource of the model, the path component of the request URI refers to the parent resource of the top level managed object instances as defined in clause 4.4.4.

|  |
| --- |
| POST /ProvMnS/1700 HTTP/1.1Host: example.orgContent-Type: application/json{ "id": "null", "objectClass": "SubNetwork", "attributes": { "userLabel": "Berlin NW", "userDefinedNetworkType": "5G", "plmnId": { "mcc": 456, "mnc": 789 } }} |

## A.3.3 Creation of multiple resources with 3GPP JSON Merge Patch

One or more resources can be created with a single 3GPP JSON Merge Patch request. The following example shows the creation of a complete subtree for a new "ManagedElement" below "SubNetwork".

The target URI has been chosen to identify the first common ancestor of the resources to be created. In this case, it is the parent of the base object of the tree to be created. The "objectClass" property is present for the resources to be created.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-merge-patch+json{ "id": "SN1", "ManagedElement": [ { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" }, "XyzFunction": [ { "id": "XYZF1", "objectClass": "XyzFunction", "attributes": { "attrA": "xyz", "attrB": 771 } }, { "id": "XYZF2", "objectClass": "XyzFunction", "attributes": { "attrA": "abc", "attrB": 772 } } ] } ]} |

<add new example>

The next example shows how a new "XyzFunction" resource is added to each of the "ManagedElement" resouces.

In this case, the parent of the parent of the "XyzFunction" resources to be created has been chosen as the common ancestor referenced by the target URI. The "objectClass" property is present for the resources to be created.

The "ManagedElement" resources are present with their "id" only. These resources are required to bridge the containment tree from the "SubNetwork" target resource to the created "XyzFunction" resources.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-merge-patch+json{ "id": "SN1", "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF3", "objectClass": "XyzFunction", "attributes": { "attrA": "def", "attrB": 553 } } ] }, { "id": "ME2", "XyzFunction": [ { "id": "XYZF1", "objectClass": "XyzFunction", "attributes": { "attrA": "def", "attrB": 661 } } ] } ]} |

|  |
| --- |
|  |

## A.3.4 Creation of multiple resources with 3GPP JSON Patch

One or more resources can be created with a single 3GPP JSON Patch request. The following example shows the creation of a complete subtree for a new network entity represented by a "ManagedElement" resource and two "XyzFunction" resources. The target URI has been chosen to identify the first common ancestor of the resources to be created. The "path" specifies the offset to the resource to be created. The "path" has no fragment component. Parent resources are created before child resources following the order of the operations in the patch document. The class name of the object to be created is specified in each patch operation.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "add", "path": "/ManagedElement=ME3", "value": { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" } } }, { "op": "add", "path": "/ManagedElement=ME3/XyzFunction=XYZF1", "value": { "id": "XYZF1", "objectClass": "XyzFunction", "attributes": { "attrA": "xyz", "attrB": 771 } } }, { "op": "add", "path": "/ManagedElement=ME3/XyzFunction=XYZF2", "value": { "id": "XYZF2", "objectClass": "XyzFunction", "attributes": { "attrA": "abc", "attrB": 772 } } }] |

Note that each resource to be created shall be specified with a dedicated "add" operation. The following patch document is hence invalid as it attempts to create three resources with a single "add" operation.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "add", "path": "/ManagedElement=ME3", "value": { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" }, "XyzFunction": [ { "id": "XYZF1", "objectClass": "XyzFunction", "attributes": { "attrA": "xyz", "attrB": 771 } }, { "id": "XYZF2", "objectClass": "XyzFunction", "attributes": { "attrA": "abc", "attrB": 772 } } ] } }] |

It is not an error if the target location of an "add" operation as specified by the "path" property does exist. In this case the content of the target location is replaced with the content of the "value" property. For example, in the following example, the first "ManagedElement" resource already exists. The patch document is applied successfully though. The representation of the first "ManagedElement" resource is replaced and the second "ManagedElement" resource is created.

Note that the attributes "vendorName" and "location" are removed from the representation of the first "ManagedElement" resource. The "userLabel" attribute is updated.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "add", "path": "/ManagedElement=ME2", "value": { "id": "ME2", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 4" } } }, { "op": "add", "path": "/ManagedElement=ME3", "value": { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" } } } ] |

# A.4 Deletion of resources

## A.4.1 Deletion of a resource with HTTP DELETE

The following example deletes an instance of "ManagedElement". The resource to be deleted is identified with the target URI. The request body is absent.

|  |
| --- |
| DELETE /SubNetwork=SN1/ManagedElement=ME2 HTTP/1.1Host: example.org |

|  |
| --- |
| HTTP/1.1 204 No ContentDate: Tue, 06 Aug 2019 16:50:26 GMT |

## A.4.2 Deletion of multiple resources with HTTP DELETE

The deletion of multiple resources with a single HTTP DELETE request is not supported. The following request is hence invalid.

|  |
| --- |
| DELETE /SubNetwork=SN1?scopeType= BASE\_NTH\_LEVEL&scopeLevel=2 HTTP/1.1Host: example.org |

|  |
| --- |
|  |

## A.4.3 Deletion of multiple resources with 3GPP JSON Merge Patch

One or more descendant resources of the target URI can be deleted with a single 3GPP JSON Merge Patch request. The following example deletes the "ManagedElement" resource with "ME1" including both its "XyzFunction" resources.

The target URI has been chosen to identify the first common ancestor of the resources to be deleted. The patch document starts with the target resource. All resources of the subtree to be deleted are marked for deletion.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-merge-patch+json{ "id": "SN1", "ManagedElement": [ { "id": "ME1", "attributes": "null", "XyzFunction": [ { "id": "XYZF1", "attributes": "null" }, { "id": "XYZF2", "attributes": "null" } ] } ]} |

|  |
| --- |
|  |

## A.4.4 Deletion of multiple resources with 3GPP JSON Patch

Multiple resources are deleted with an ordered sequence of "remove" operations. The following example removes a complete subtree for a "ManagedElement".

The target URI has been chosen to identify the parent resource of the "ManagedElement" resource to be deleted. The "path" specifies the offset to the resources to be deleted. The "path" has no fragment component.

Child resources are deleted before parent resources, starting with leaf resources.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "remove", "path": "/ManagedElement=ME1/XyzFunction=XYZF1" }, { "op": "remove", "path": "/ManagedElement=ME1/XyzFunction=XYZF2" }, { "op": "remove", "path": "/ManagedElement=ME1" }] |

# A.5 Complete update of a resource

The following example updates a "XyzFunction" resource. Only the "attrA" attribute is updated with a new value "def". The "attrB" attribute is set to the old value "551", but still the "attrB" attribute needs to be present in the resource representation contained in the request message body. Otherwise "attrB" would be deleted due to the replace semantics of HTTP PUT.

|  |
| --- |
| PUT /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1 HTTP/1.1Host: example.orgContent-Type: application/json{ "id": "XYZF1", "attributes": { "attrA": "def", "attrB": 551 }} |

When a non leaf resource is updated, contained resources are not included. For example, the following resource representation in the message body updates the "ManagedElement" resource only. It does not delete the contained "XyzFunction" resources.

|  |
| --- |
| PUT /SubNetwork=SN1/ManagedElement=ME1 HTTP/1.1Host: example.orgContent-Type: application/json{ "id": "ME1", "attributes": { "userLabel": "Berlin New Label", "vendorName": "Company XY", "location": "TV Tower" }} |

# A.6 Partial update of a resource

## A.6.1 Partial update of a resource with JSON Merge Patch

The first example shows how the attribute "attrA" of the "XyzFunction with the "id" equal to "YXZF1" is changed from "xyz" to "def" using JSON Merge Patch.

|  |
| --- |
| PATCH /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1 HTTP/1.1Host: example.orgContent-Type: application/merge-patch+json{ "id": "XYZF1", "attributes": { "attrA": "def" }} |

In the second example the "mcc" attribute field of the "plmnId" attribute is updated to "654". The employed patch method is again JSON Merge Patch.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/merge-patch+json{ "id": "SN1", "attributes": { "plmnId": { "mcc": 654 } }} |

In the third example the item "Metric3" is added to the array "perfMetrics". The value of "perfMetrics" contains the two old items and the new item.

|  |
| --- |
| PATCH /SubNetwork=SN1/PerfMetricJob=PMJ1 HTTP/1.1Host: example.orgContent-Type: application/merge-patch+json{ "id": "PMJ1", "attributes": { "perfMetrics": ["Metric1", "Metric2, Metric3"] } }} |

Also in case the items of an array have an identifier, the complete updated array value needs to be present in the patch request. In the following fourth example in this clause the old first threshold level is deleted, for the old second threshold level the "value" is updated from "20" to "22", the old third threshold level is left unchanged, and a new threshold level is appended as last item.

|  |
| --- |
| PATCH /SubNetwork=SN1/ThresholdMonitor=TM1 HTTP/1.1Host: example.orgContent-Type: application/merge-patch+json{ "id": "TM1", "attributes": { "thresholdLevels": [ { "level": "2", "thresholdValue": 22 }, { "level": "3", "thresholdValue": 30 }, { "level": "4", "thresholdValue": 40 } ] }} |

## A.6.2 Partial update of a resource with 3GPP JSON Merge Patch

|  |
| --- |
|  |

|  |
| --- |
|  |

When updating a single resource, there is no difference between JSON Merge Patch (see A.6.1) and 3GPP JSON Merge Patch.

## A.6.3 Partial update of a resource with JSON Patch

When JSON Patch is used to request the same changes as the ones described in the four examples in clause A.6.1, the MnS consumer may send

|  |
| --- |
| PATCH /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1 HTTP/1.1Host: example.orgContent-Type: application/json-patch+json[ { "op": "replace", "path": "/attributes/attrA", "value": "def" }] |

and

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/json-patch+json[ { "op": "replace", "path": "/attributes/plmnId/mcc", "value": 654 }] |

and

|  |
| --- |
| PATCH /SubNetwork=SN1/PerfMetricJob=PMJ1 HTTP/1.1Host: example.orgContent-Type: application/json-patch+json[ { "op": "add", "path": "/attributes/perfMetrics/2", "value": "Metric3" }] |

and

|  |
| --- |
| PATCH /SubNetwork=SN1/ThresholdMonotor=TM1 HTTP/1.1Host: example.orgContent-Type: application/json-patch+json[ { "op": "remove", "path": "/attributes/thresholdLevels/0" }, { "op": "replace", "path": "/attributes/thresholdLevels/0/thresholdValue", "value": 22 }, { "op": "add", "path": "/attributes/thresholdLevels/-", "value":  { "level": "4", "thresholdValue": 40 } }] |

Note the patch operations are applied sequentially to the "thresholdLevels" array in the order they appear in the patch array. After removing the first array item with the first operation, the resulting array value becomes the target for the second operation. The array index "0" identifies the new first item, which was the second item before applying the first operation of the patch document. Issues with array positions can be avoided by placing "replace" operations at the beginning of the patch document.

## A.6.4 Partial update of a resource with 3GPP JSON Patch

When 3GPP JSON Patch is used to request the changes described in the first two examples in clause A.6.1 the MnS consumer may send the following

|  |
| --- |
| PATCH /SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "replace", "path": "#/attributes/attrA", "value": "def" }] |

and

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "replace", "path": "#/attributes/plmnId/mcc", "value": 654 }] |

and

|  |
| --- |
| PATCH /SubNetwork=SN1/ThresholdMonitor=TM1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "remove", "path": "#/attributes/thresholdLevels/0" }, { "op": "replace", "path": "#/attributes/thresholdLevels/0/value", "value":  { "thresholdValue": 22 } }, { "op": "add", "path": "#/attributes/thresholdLevels/-", "value":  { "level": "4", "thresholdValue": 40 } }] |

When using 3GPP JSON Patch to update a single resource, the only difference compared to JSON Patch is the presence of "#" in the "path".

# A.7 Manipulating multiple resources

## A.7.1 Manipulating multiple resources with 3GPP JSON Merge Patch

JSON Merge Patch allows to update one resource only with a single HTTP PATCH request. The resource must exist. In contrast, 3GPP JSON Merge Patch allows to update multiple resources incl. resource creation and deletion with a single HTTP PATCH request.

In the following example the "userLabel" attribute and the "mcc" attribute field of the "SubNetwork" resource are updated. The "attrB" attribute of the "XyzFunction" resource, whose "id" is "XYZF1", is also updated. A new "XyzFunction" resource with id "XYZF3"is created as well as a new "ManagedElement" resource with id "ME3". The "XYzFunction" resource, whose "id" is "XYZF2", is deleted.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-merge-patch+json{ "id": "SN1", "attributes": { "userLabel": "Berlin NW-1", "plmnId": { "mcc": 654 } }, "ManagedElement": [ { "id": "ME1", "XyzFunction": [ { "id": "XYZF1", "attributes": { "attrB": 1234 } }, { "id": "XYZF2", "attributes": "null" }, { "id": "XYZF3", "objectClass": "XyzFunction", "attributes": { "attrA": "fgh", "attrB": 555 } } ] }, { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" } } ]} |

|  |
| --- |
|  |

## A.7.2 Manipulating multiple resources with 3GPP JSON PATCH

|  |
| --- |
| The same resource modifications as in the previous clause expressed using 3GPP JSON Patch are given byPATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "replace", "path": "#/attributes/userLabel", "value": "Berlin NW-1" }, { "op": "replace", "path": "#/attributes/plmnId/mcc", "value": 654 }, { "op": "replace", "path": "ManagedElement=ME1/XyzFunction=XYZF1#/attributes/attrB", "value": 1234 }, { "op": "add", "path": "/ManagedElement=ME1/XyzFunction=XYZF3", "value": { "id": "XYZF3", "objectClass": "XyzFunction", "attributes": { "attrA": "ghi", "attrB": 553 } } }, { "op": "remove", "path": "/ManagedElement=ME1/XyzFunction=XYZF2" }, { "op": "add", "path": "/ManagedElement=ME3", "value": { "id": "ME3", "objectClass": "ManagedElement", "attributes": { "userLabel": " Berlin NW 3", "vendorName": "Company XY", "location": "Spandau" } } }] |

The two modifications of the "userLabel" attribute and the "mcc" attribute filed can also be expressed by a single "merge" operation rather than two separate "replace" operations using 3GPP JSON Patch:

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1Host: example.orgContent-Type: application/3gpp-json-patch+json[ { "op": "merge", "path": "#/attributes", "value": { "userLabel": "Berlin NW-1", "plmnId": { "mcc": 654 } } }] |

|  |
| --- |
| **End of modifications** |