|  |  |
| --- | --- |
| 3GPP TR 28.831 V18.0.0 (2023-06) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Study on basic Service-Based Management Architecture  (SBMA) enabler enhancements  (Release 18) | |
|  | |
|  |  |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2023, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 5

1 Scope 7

2 References 7

3 Definitions of terms, symbols and abbreviations 8

3.1 Terms 8

3.2 Symbols 8

3.3 Abbreviations 8

4 Issues 8

4.1 Void 8

4.2 Key issue #2: Targeted notification subscriptions 8

4.2.1 Issue description 8

4.2.2 Current situation 8

4.2.2.1 Scoping objects based on object classes 8

4.2.2.2 Scoping attributes 10

4.2.2.3 Scoping objects or attributes based on conditions 11

4.2.3 Analysis 11

4.2.4 Potential requirements 11

4.2.5 Potential solutions 12

4.2.5.1 Introduction 12

4.2.5.2 XPath 1.0 12

4.2.5.2.1 General considerations 12

4.2.5.2.2 Special considerations for JSON 14

4.2.5.2.3 Special considerations for YANG 17

4.2.5.3 XPath 2.0 17

4.2.5.2.4 XPath 1.0 profiles 17

4.2.5.4 XPath 3.1 19

4.2.5.5 JSON Pointer 19

4.2.5.6 JSONPath 19

4.2.6 CR proposal 20

4.2.7 Conclusion 23

4.3 Issue #3: Definition of createMOI 23

4.3.1 Issue description 23

4.3.2 Analysis 23

4.3.3 CR proposal 24

4.3.4 Conclusion 25

4.4 Issue #4: Definition of modifyMOIAttributes 25

4.4.1 Issue description 25

4.4.2 Analysis 25

4.4.3 CR proposal 25

4.4.4 Conclusion 28

4.5 Issue #5: HTTP error response format 28

4.5.1 Issue description 28

4.5.2 Analysis 28

4.5.3 CR proposal 28

4.6 Issue#6: Operation for multiple MOI updates (stage 2) 46

4.6.1 Issue description 46

4.6.2 Analysis 46

4.6.3 CR proposal 46

4.6.4 Conclusion 47

4.7 Issue#7: Advertising communication options (REST SS) 47

4.7.1 Issue description 47

4.7.2 Analysis 47

4.7.3 CR proposal 48

4.7.4 Conclusion 50

4.8 Issue #8: Logging 50

4.8.1 Issue description 50

4.8.2 Analysis 50

4.8.3 Potential Requirements 50

4.8.4 Potential Solutions 50

4.8.5 CR proposal 50

4.9 Issue#9: Harmonized approach for node selection 51

4.9.1 Issue Description 51

4.9.2 Analysis 51

4.9.2.1 Node selection in read operations 51

4.9.2.2 Node selection with NRM attributes 51

4.9.3 Potential requirements 52

4.9.4 Potential solution 52

4.9.5 Conclusion 52

4.10 Issue#10: OAS definition versioning (HTTP/JSON only) 52

4.10.1 Issue Description 52

4.10.2 Analysis 53

4.10.3 Potential requirements 53

4.10.4 Potential solutions 53

4.10.5 CR proposal 53

4.10.6 Conclusion 53

4.11 Issue #11: Advertising NRM properties by MnS producer 53

4.11.1 Issue description 53

4.11.2 Current situation 54

4.11.3 Analysis 55

4.11.4 Potential requirements 55

4.11.5 Potential solution 55

4.11.6 CR proposal 56

4.11.7 Conclusion 56

4.12 Issue #12: Multiple alarms subscription mechanism 56

4.12.1 Issue description 56

4.12.2 Analysis 56

4.12.3 Potential Requirements 56

4.12.2 Potential Solutions 56

4.12.3 Change Request proposal 56

4.13 Issue #13: Partial success for update operations (HTTP/JSON only) 57

4.13.1 Issue description 57

4.13.2 Analysis 57

4.13.3 Potential requirements 57

4.13.4 Potential solutions 57

4.13.5 CR proposal 58

4.13.6 Conclusion 58

5 Recommendations and conclusions 58

Annex A: Profiles 58

A.1 EBNF for basic XPath profile 58

A.2 EBNF for advanced XPath profile 58

Annex B: Change history 61

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document …

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999 (Status updated October 2016), (https://www.w3.org/TR/1999/REC-xpath-19991116/)

[3] XML Path Language (XPath) 2.0, W3C Recommendation 14 December 2010 (Link errors corrected 3 January 2011; Status updated October 2016), (https://www.w3.org/TR/xpath20/)

[4] XML Path Language (XPath) 3.1, W3C Recommendation 21 March 2017 (<https://www.w3.org/TR/xpath-31/>)

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

[6] IETF Internet-Draft: "JSONPath: Query expressions for JSON"; draft-ietf-jsonpath-base-05; April 2022 (<https://datatracker.ietf.org/doc/html/draft-ietf-jsonpath-base-05>).

[7] XML Information Set (Second Edition), W3C Recommendation 4 February 2004 (<https://www.w3.org/TR/xml-infoset/>)

[8] XQuery and XPath Data Model 3.1, W3C Recommendation 21 March 2017 (<https://www.w3.org/TR/xpath-datamodel-31/>)

[9] IETF RFC 7950: "The YANG 1.1 Data Modeling Language"

[10] XForms 2.0 (https://www.w3.org/community/xformsusers/wiki/XForms\_2.0)

[11] 3GPP TS 32.158: "Management and orchestration; Design rules for REpresentational State Transfer (REST) Solution Sets (SS)"

[12] IETF RFC 787: "Problem Details for HTTP APIs."

[13] 3GPP TS 28.537: “Management and orchestration; Management capabilities”.

[14] 3GPP TS 28.662: "Telecommunication management; Generic Radio Access Network (RAN) Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS) ".

[15] 3GPP TS 28.533: “Management and orchestration; Architecture framework”.

[16] RFC8525: YANG Library

[17] 3GPP TS 32.160: “Management and orchestration; Management service template”.

[18] IETF RFC 7950: “The YANG 1.1 Data Modeling Language, August 2016”.

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

**example:** text used to clarify abstract rules by applying them literally.

## 3.2 Symbols

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

<symbol> <Explanation>

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

<ABBREVIATION> <Expansion>

# 4 Issues

## 4.1 Void

## 4.2 Key issue #2: Targeted notification subscriptions

### 4.2.1 Issue description

SA5 is moving to a fully model driven approach. In this architecture all aspects that are managed are represented in the information model. It is possible to observe all changes in the network by observing the changes in the information model. Note the information model covers all fragments: configuration management, alarm management and performance management.

Many MnS consumers are not interested in all changes in the network. For that reason, it is important that MnS consumers can subscribe to specific portions of the information model only. These portions may be described in terms of object instances, attributes or attribute fields.

The currently specified "NtfScubscriptionControl" does not allow to scope subscriptions with that granularity. Some enhancements seem to be required to "NtfScubscriptionControl". This clause analyses the current situation and proposes a solution.

### 4.2.2 Current situation

#### 4.2.2.1 Scoping objects based on object classes

For scoping managed objects, the attribute "scope" is provided in "NtfSubscriptionControl". It has two attribute fields: "scopeType" and "scopeLevel". This scoping mechanism works only on the hierarchical levels of the information model. For example, you can select the level below "SubNetwork".

Typically, instances of more than one object class can be located on a level. For example, below "SubNetwork" there can be instances of "ManagedElement", "PerfMetricJob", "TraceJob" and "AlarmList". It is not possible to scope only instances of one or more specific object classes with the current scoping mechanism.

To reduce the scoped set of object instances to those with a specific object class, the "notificationFilter" attribute can be used. The IS level parameters "objectClass" and "objectInstance" are typically mapped to one stage 3 parameter ("href") only. There is no dedicated parameter for the "objectClass" in stage 3. Filtering out notifications related to certain object classes involves therefore the application of string functions on "href".

The described mechanism applies to CM notifications and non-CM notifications such as alarm notifications.

Note on all examples:

The following examples are for the RESTful HTTP-based solution only. For this solution the "notificationFilter" format is XPath 1.0 in Rel-17. In addition, the JSON instance document sent over the wire is expressed in XML. A <notification> element is added as root element to produce a valid XML document, a necessary manipulation not described yet in Rel-17.

**Example 1 (alarm notifications):**

Assume the NRM specified in TS 28.622 is supported on a MnS producer. A subscription scoping (with "scopeType" and "scopeLevel") the level below "SubNetwork" has been created. No notification filter is specified. The subscription is for alarm notifications only. On the level below "SubNetwork" there are instances of "ManagedElement", "PerfMetricJob" and "NtfSubscriptionControl".

The described subscription forwards alarm notifications related to all manged object classes. For example, all following notifications are forwarded:

<notification>

<href>example.com/SubNetwork=SN1/ManagedElement=ME1</href>

...

</notification>

<notification>

<href>example.com/SubNetwork=SN1/ManagedElement=ME2</href>

...

</notification>

<notification>

<href>example.com/SubNetwork=SN1/PerfMetricJob=PMJ1</href>

...

</notification>

<notification>

<href>example.com/SubNetwork=SN1/NtfSubscriptionControl=NSC1</href>

...

</notification>

The following notification is not related to the set of scoped objects and hence not forwarded:

<notification>

<href>example.com/SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1</href>

...

</notification>

The subscription shall be modified now to include only "ManagedElement" objects. This is realized with an appropriate XPath expression as value of "notificationFilter". Possible XPath expressions include:

**XPath expression 1:**

The following XPath expression returns the "notification" node since the string "ManagedElement" is contained in the "href" value.

"/notification[contains(href,"ManagedElement")]"

**XPath expression 2:**

The following expression checks for the presence of "ManagedElement" after "example.com/SubNetwork=SN1".

"/notification[starts-with\

(substring-after(path,"example.com/SubNetwork=SN1/"),\

"ManagedElement")]"

**Example 2 (alarm notifications):**

Assume again the NRM specified in TS 28.622 is supported on a MnS producer. "PerfMetricJob" instances have been created below "SubNetwork" and below some "ManagedElement" instances. A MnS consumer wishes to create a subscription related to all "PerfMetricJob" instances.

The notification scope needs to include the complete object tree starting at "SubNetwork". In case a "PerfMetricJob" instance is always a leaf object, then checking for the presence of the sub-string "PerfMetricJob" in "href" does the job. In case a "PerfMetricJob" instance is not always a leaf object, then it is necessary to check, if the last path segment of "href" identifies a "PerfMetricJob". A simple test on if the "href" value includes the sub-string "PerfMetricJob" is not sufficient. There is no obvious Xpath expression to solve this problem, at least not with XPath1.0.

**Example 3 ("notifyMOICreation", "notifyMOIDeletion", "notifyMOIAttributeValueChanges")**

The created, dfeleted or updated objerct is specified with the "href" parameter of the notification header. For that reason the same considerations as for alarm notifications apply.

**Example 3 ("notifyMOIChanges")**

The objects are identifierd with both the "href" and "path" parameter. Filtering on these parameters to reduce the scoped set of objects to the sub-set of interest is not a feasible solution.

#### 4.2.2.2 Scoping attributes

The scoping mechanism allows to select a set of objects. This is good enough for notification types related to the complete object, such as alarm notifications, or when a MnS consumer is interested in receiving attribute value change notifications for all attributes of an object. It is not possible to target specific attributes of an object only using the scoping mechanism. When a MnS consumer is interested only in value changes of one or more specific attributes of an object, then the MnS consumer needs to configure into the subscription an appropriate notification filter.

Note the following examples are for the RESTful HTTP-based solution only.

**Example 1 ("notifyMOIAttributeValueChanges"):**

Assume a subscription for "notifyMOIAttributeValueChanges" includes the instance of "XyzFunction" identified by "XYZF1". A notification reporting the value change of "attrA" and "attrB" looks like

<notification>

<href>example.com/SubNetwork=SN1/ManagedElement=ME1/XyzFunction=XYZF1</href>

...

<attributeListValueChanges>

<attrA>123</attrA>

<attrB>abc</attrA>

</attributeListValueChanges>

<attributeListValueChanges>

<attrA>456</attrA>

<attrB>def</attrA>

</attributeListValueChanges>

</notification>

If a MnS consumer is interested only in value changes of "attrA", then he needs to specify a notification filter testing on the presence of "attrA", for example:

"/notification/attributeListValueChanges[attrA]"

Note that the complete notification including also "attrB" is forwarded in case the test evaluates to true. The "attrB" cannot be removed from the notification. With filtering either the complete notification is forwarded, or the complete notification is not forwarded.

Note also that for "notifyMOIChanges" the situation is more complicated. Changes of multiple managed object instances can be reported using a single "notifyMOIChanges" notification. Even worse, the object instance in the notification header may not even be an instance that has changed, but only a common ancestor of the instances that have changed. This means that not only the "href" property of the notification header needs to be checked but also the "path" properties of the notification body.

#### 4.2.2.3 Scoping objects or attributes based on conditions

It is currently not possible to specify a conditional scope. A conditional scope could be based for example on the presence of an attribute or on an attribute with a specific value. Note that scoping based on object classes can be considered as conditional scope.

For example, a MnS consumer might be interested only in changes of "AlarmInformation" instances (reported with alarm notifications) with a perceived severity equal to "CRITICAL".

### 4.2.3 Analysis

Scoping capabilities are currently limited.

### 4.2.4 Potential requirements

Potential requirements to address the issues identified above are

[Req-1] A subscription shall enable to select a single object instance.

[Req-2] A subscription should enable to select a complete subtree of objects starting at a specified base object.

[Reg-3] A subscription shouldenable to select the objects on a specified level below a specified base object.

[Req-4] A subscription shouldenable to select the objects starting at a specified base object down to and including a specified level below the base object.

[Req-5] A subscription shouldenable to select multiple object instances based on object instance identifiers.

[Req-6] A subscription shouldenable to select multiple object instances based on object classes.

[Req-7] A subscription shouldenable to select multiple object instances based on filter conditions.

The following potential additional requirements apply for subscriptions to CM notifications:

[Req-8] A subscription shouldenable to select attributes within selected object instances.

[Req-9] A subscription shouldenable to select attribute fields within selected object instances.

[Req-10] A subscription shouldenable to select attribute elements within selected object instances.

[Req-11] A subscription shouldenable to select attributes, attribute fields and attribute elements based on filter conditions.

Note only [Req-1] is mandatory. All other requirements are optional. This allows for implementations of different complexity that are tailored for different deployment needs.

The MnS consumer should be able to get the information which exact subscription capabilities are supported by a MnS producer, or more specifically by "NtfSubscriptionControl". This may require the introduction of capability identifiers that can be retrieved by MnS consumers.

### 4.2.5 Potential solutions

#### 4.2.5.1 Introduction

Solutions to the potential requirements in clause 4.2.4 need to be able to conditionally select nodes in node trees.

Note: The NRM is a node tree. Node types are objects and attributes. The tree is defined by the name-containment relationships.

The target is to use an existing notation as solution, ideally without any modifications. The following notations are analyzed for use with JSON defined NRMs:

* XPath 1.0 [2]
* XPath 2.0 [3]
* XPath 3.1 [4]
* JSON Pointer [5]
* JSONPath [6]

The following notations are analyzed for use with YANG defined NRMs:

* XPath 1.0 [2]
* XPath 2.0 [3]
* XPath 3.1 [4]

All notations listed above are quite powerful. Only a subset is required to support the potential requirements documented in clause 4.2.4. This subset is identified in the following clauses.

*Editor's note: It is ffs if a stage 2 definition is required, that is more detailed than the requirements, and from which the subset can be identified in a direct mapping.*

*Editor's note: XPath does not work on XML. Instead, it works on an own data model defined in XPath 1.0 [2], clause 5. This clause also describes the mapping of an XML document to the XPath model. The mapping of a JSON document to the XPath model is ffs. Alternatively, some mapping between a JSON document to a XML document needs to be looked at.*

#### 4.2.5.2 XPath 1.0

##### 4.2.5.2.1 General considerations

XPath 1.0 [2] has been designed primarily to select one or more nodes of an XML document. However, XPath is a notatition that is not tied to XML and "operates on the abstract, logical structure of an XML document or JSON object, rather than its surface syntax.", as clarified in XML Path Language (XPath) 3.1 [4], clause 1. This is because an XPath expression operates on documents with an own XPath specific data model, and not on the XML or JSON surface syntax. This data model is specified in XPath 1.0 [2], clause 5. Note the data model for XPath 2.0 [3] and XPath 3.1 [4] is described in XQuery and XPath Data Model 3.1 [8].

Since an XPath expression works on an own data model, it is necessary to translate XML of JSON input documents into that data model.

For XML this translation is clearly defined by W3C. It is a two-step process. The mapping from the information in an XML document to the XML Information Set is described in XML Information Set [7]. The mapping from the XML Information Set to the XPath data model is described in XPath 1.0 [2], annex B. The mapping from YANG to XML is defined in RFC 7950 [9].

However, W3C did not define a clear mapping of a JSON document into the XPath data model, see the following clause for more details.

XPath models an XML or JSON document as a tree of nodes. NRM objects and NRM attributes are both mapped to a specific node type of the XPath data model, the element node. XPath 1.0 does hence not distinguish between element nodes representing NRM objects and element nodes representing NRM attributes.

Furthermore, XPath 1.0 does not distinguish between child element nodes that represent NRM attributes and those child nodes (at the same level in the the tree hierarchy) that represent name contained objects.

These two observations have some important implications when using XPath 1.0 for selecting NRM objects:

The first implication is that it is not possible to select all NRM attributes of an NRM object unless some measures are taken. Assume the XPath expression "SubNetwork/\*", it selects all attributes of SubNetwork, but also the name contained objects. Fortunately, the attributes are embedded in an "attributes" container in the YANG and JSON defined NRMs. Therefore, it is easy to construct an XPath expression selecting all attributes of an object. The XPath expression is "SubNetwork/attributes" (when the "attributes" element node and the attribute element nodes are selected) or "SubNetwork/attributes/\*" (when only the attribute element nodes are selected).

The second implication is that when selecting the child objects of an object, for example with the XPath expression "SubNetwork/ManagedElement", which selects all "ManagedElement" childs of "SubNetwork", then also the objects name contained by the "ManagedElement" childs are selected, and so forth. In other words, the child axis works based on element nodes and not on NRM objects as desired. XPath 1.0 does not offer any solution for this problem. Only the attributes container may be used again to select only the attributes of the "ManagedElement" childs, the XPath expression to select the "ManagedElement" childs is hence "SubNetwork/ManagedElement/attributes".

Note XPath 1.0 is verbose and does not allow for compact expressions in many cases. For example, to select a subset of all attributes of an object, each attribute needs to be selected individually with the complete location path:

/SubNetwork/attributes/attrA | /SubNetwork/attributes/attrB

And when multiple objects shall be selected based on object instance names, then the complete location path and predicate needs to be repeated for each instance

//\*[objectInstance="DN1"]/attributes | //\*[objectInstance= "DN2"]/attributes

An XPath expression is evaluated within a context. The context consists of:

* a node (the context node)
* a pair of non-zero positive integers (the context position and the context size)
* a set of variable bindings
* a function library
* the set of namespace declarations in scope for the expression

The context needs to be clearly defined.

**XPath expression examples (assuming the presence of an "attributes" container in the info model):**

Select the specified base object

/\*/attributes

Select all objects in the tree starting at the specified base object:

//attributes

Select all objects on a specific level below a specified base object, in this case the second level below the base object:

/\*/\*/\*/attributes

Select the objects starting from a specified base object down to and including the objects on a specific level below the base object:

/\*/attributes | /\*/\*/attributes | /\*/\*/\*/attributes

Select objects based on their object instance identifiers in the tree starting at the base object:

//\*[objectInstance="DN1"]/attributes | //\*[objectInstance= "DN2"]/attributes

Select objects based on their object class

/SubNetwork/attributes

/SubNetwork[id="SN1"]/ManagedElement/attributes

//\*[objectClass="NtfSubscriptionControl"]/attributes

Select specific attributes and attribute fields:

<objectSelector>/attrA | <objectSelector>/attrB/subAttrB1

<objectSelector>/attributes/attrA | <objectSelector>attributes/attrB/subAttrB1

Select objects and attributes based on predicates (conditions):

<objectSelector>[<predicate>]

<objectSelector>/attributes<attributesSelector>[<predicate>]

##### 4.2.5.2.2 Special considerations for JSON

As stated in the previous clause W3C did not provide a real mapping from JSON to the XPath data model. It is possible though to use a mapping from JSON to XML defined in XForms2.0 [10], clause 5.2.2. The resulting XML document can then be mapped to the XPath data model.

Note, XForms2.0 is a W3C draft from 2010 and not a W3C standard. The produced XML document contains in XML attributes information about the original JSON data taypes. JSON arrays are mapped in a very specific way to XML (see example below). These mapping rules preserve all information for mapping the XML document again back to the original JSON document.

For example, the JSON snippet

"load": [0.31, 0.33, 0.32]

is mapped according to XForms2.0 to

<load type="array">

<\_ type="number">0.31</\_>

<\_ type="number">0.33</\_>

<\_ type="number">0.32</\_>

</load>

Other best practices propose to map the JSON snippet as follows:

<load>0.31</load>

<load>0.33</load>

<load>0.32</load>

There is hence not one and only one standard specifying the mapping from a JSON document to an XML document. One could argue that the XML document is anyway a conceptual document only that is not visible on the wire and used only internally in the MnS producer helping to apply the XPath expression to a JSON document. However, the exact way how JSON is mapped to XML has implications on the XPath expression to select nodes.

For example, when the first mapping method is used, the XPath expression snippet to select the first array item is

/load/\_[1]

When the second mapping method is used, the XPath expression snippet is

/load/[1]

For a really interoperable solution, the MnS consumer needs to know the exact way the MnS producer maps JSON to XML. This means in turn, that the mapping method needs to be standardized by SA5.

A second option is to provide a mapping from the information elements in a JSON document to the XPath data model directly without an XML mapping in-between. Such a mapping is not provided at all by W3C.

The XPath data model has seven types of nodes: root nodes, element nodes, text nodes, attribute nodes, namespace nodes, processing instruction nodes, comment nodes. Each node has multiple properties.

Here after, a proposal to directly map a JSON document to the XPath data model:

* The JSON document is mapped to the root node.
* The name of a name/value pair is mapped to an element node. The element node coming from the name is a child of the root node. Vice versa, the root node is the parent of the element node coming from the name.
* The value of a name/value pair is mapped to a text node if the value is a string, a number, or one of the tree literal names: false, true, null. The element node coming from the name is the parent of the text node coming from the value. Vice versa, the text node coming from the value is the child of the element node coming from the name.
* The value of a name/value pair is mapped to one or more element nodes if the value is an object. There is one element node for each name/value pair contained in the object. The names of the element nodes are equal to the names of the name/value pairs the element nodes are coming from. The element node coming from the name is the parent of the element nodes coming from the value. Vice versa, the element nodes coming from the value are the children of the element node coming from the name.
* The value of a name/value pair is mapped to one or more element nodes if the value is an array. There is one element node for each array item contained in the array. The names of the element nodes are all identical and equal to the name of the name/value pair they are coming from. The element node coming from the name is the parent of the element nodes coming from the value. Vice versa, the element nodes coming from the value are the children of the element node coming from the name.

We can conclude that only root nodes, element nodes and text nodes are used. Attribute nodes, namespace nodes, processing instruction nodes, and comment nodes are not used.

As to node properties, only name, children and parent are used.

In case XPath is chosen as node selection language, then it is necessary to specify in normative fashion either the JSON to XML mapping, or the JSON to XPath data model mapping.

Now we will look at if due to some inherent properties of JSON all XPath concepts are applicable, when the original document from which nodes are selected, is a JSON document.

A JSON object is an unordered collection of zero or more name/value pairs. This is why the concept of document order (clause 5 of XPath 1.0 [2]) is not applicable when an XPath expression is applied to an XPath data model that was generated from a JSON document. This has the following implications:

* The following axes cannot be used: following, following-sibling, preceding, preceding-sibling.
* The following functions cannot be used: position, local-name, namespace-uri, name.

JSON does not have a namespace concept. This is why XPath concepts related to name spaces are not applicable. This has the following implications:

* The following axes cannot be used: namespace
* The following functions cannot be used: local-name, namespace-uri, name

JSON does not have a concept similar to XML attributes. This is why XPath concepts related to attributes are not applicable. This has the following implications:

* The following axes cannot be used: attribute

Out of the seven node types specified in XPath 1.0 [2] only the root node, element node and text node are used. The attribute nodes, namespace nodes, processing instruction nodes and comment nodes are not used.

The name of the root element node is the class name of the base object. The base object is the node that contains the "NtfSubscriptionControl" instance (that in turn has an attribute whose value is the XPath expression).

Note that the root element node (document element) is not the same as the root node. The root element node is the mapping of the top-level name/value pair in the JSON document, whereas the root node is the (conceptual) parent of that object. The root node is the mapping of the JSON document.

When the value of the top-level name/value pair is an array, which is always the case for JSON defined NRMs in SA5, this array can contain only one item, which is the base object, in the special context of notification subscription. Considerations on how to handle the case where a top-level array can contain multiple items are hence not required.

For example, assume the information model is described by annex A.1 in TS 32.158 [11]. Further assume that the "ManagedElement" with the id "ME1" contains a "NtfSubscriptionControl" instance for which a node selection XPath expression shall be constructed. The base object is the "ManagedElement" with the id "ME1". The JSON document, to which the XPath expression is applied to, is as follows:

|  |
| --- |
| {  "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  }  }  ]  }  ]  } |

or

|  |
| --- |
| {  "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  }  }  ]  } |

Since a well formed XML document has one and only one root elemet the first alternative is preferred. It contains the key "ManagedElement" that maps to the name of the XML root element. The XPath expression to identify the base object only is then

/MangedElement/attributes

or

/\*/attributes

##### 4.2.5.2.3 Special considerations for YANG

The mapping from YANG to XML is defined in RFC 7950 [9]. No special considerations are required.

#### 4.2.5.3 XPath 2.0

XPath 2.0 [3] introduces sequence expressions which help to render Xpath expressions more compact. For example, multiple attributes of an object can be selected by

/SubNetwork/(attrA, attrB)

and multiple objects based on instance names with

//\*[objectInstance=("DN1", "DN2")]/attributes

Other concepts added in XPath 2.0 like quantified expressiuons provide no obvious value for the use cases analysed.

##### 4.2.5.2.4 XPath 1.0 profiles

*Editor's note: The content of this clause is work in progress and subject to change.*

An XPath expression matches the production "Expr" defined in XPath 1.0 [2], clause 3.1. However, the general expression is much too generic for selecting nodes of an input document. For example, it allows also expressions like

(5, 256)[2]

which selects the second item in the sequence (5, 256), hence 256. This expression does not work at all on an input document. Even if "5" and "256" is replaced by some XPath expression evaluating to numbers, the expression cannot be used for selecting nodes.

This is why only one or more XPath profiles are needed.

Two XPath profiles are proposed:

* Basic profile: Supporting limited features allowing XPath to browse the document from one element node to another. The XPath EBNF is detailed in annex A.1. In this profile a location path is defined as an absolute location path. An absolute location path consists of a sequence of one or more location steps separated by / and preceded by /. The location steps in an absolute location path are composed together from left to right. Each step-in turn selects a set of nodes relative to a context node. Note that a / by itself selects the root node of the document. The basic profile supports a predicate that filters on the "id".

Example: /SubNetwork[id="SN1"]/ManagedElement[id="ME2"]/attributes

A location step is composed of

* + an axis specifier, which specifies the tree relationship between the nodes selected by the step and the context node
  + a node name which specifies the node name of the node selected by the location step. The node name can be a wildcard "\*".

The axis specifier includes two axes:

* + Child: axis containing the children of the context node

Example (unabbreviated syntax): /child::SubNetwork/child::\*

Example (abbreviated syntax): /SubNetwork/\*

* + Descendant: axis containing the descendants of the context node; a descendant is a child or a child of a child and so on

Example: /SubNetwork/descendant::\*

The predicate

* + is an equality expression with the "=" operator, the relative location path "id" on the left side and a literal string on the right side.

Note that the axis specifier "child::" can be omitted from a location step, because child is the default axis. For example, a location path /SubNetwork/ManagedElement is short for /child::SubNetwork/child::ManagedElement.

Regarding the asterisk character "\*", it is reserved to denote a wildcard when used in the location path. It selects all element children of the context node. In the the EBNF notation, the character "\*" mentions repetition symbol (it can be also represented inside curly brackets followed by the "\*").

* Advanced profile: supporting more advanced features like the usage of the predicates. The XPath EBNF is detailed in annex A.2. In this profile, a relative location path is added as a second option to the location path. A relative location path consists of a sequence of one or more location steps separated by / and it does not need to start from the root node as the absolute path.

Example: / / attributes

In this profile, in addition to the axis specifier and Node Name (same as the basic profile), a more sophisticated predicate option is added to the location step. The predicate uses arbitrary expressions to further refine the set of nodes selected by the step. Predicates are defined by an expression (PredicateExpr) always embedded in square brackets. A PredicateExpr can be as follows:

* + An EqualityExpr: expression evaluated by comparing the objects that result from evaluating the two operands. On the right side, the operand is a literal string object and on the left side a PathExpr is introduced as an operand. The PathExpr might be a location path returning an object having one of the following types: node-set, boolean, number or string.

If the PathExpr object is a node-set, then each node in the node-set is compared to the literal string defined in the right side. Note that a node in the node-set might have one/many descendants nodes. And in such a case the comparison expression is evaluated to "false". This kind of comparison is accepted by Xpath (correct syntax); and it returns an empty result. Unfortunately, in the EBNF the PathExpr can not be constrained to avoid such a comparison (See more details on handling comparison based on the node-set in clause 3.4 [2]).

Examples: /SubNetwork/ManagedElement[id="ME1"]

* + InEqualityExpr: expression defining a comparison of a PathExpr object to a given number. Here again, the same problem related to node-set comparison as discussed above is true. Note that here object to be compared is converted to a number as if by applying the number function (more details in clause 3.5 [2]).

Example: //attributes/ThresholdLevels[level>1]

* + A functionCall: evaluated by using the Function Name to identify a function in XPath function librairy. Each function in the function library is specified using a function prototype, which gives the return type, the name of the function, and the type of the arguments (more details in Clause 4[2]).

Examples:

/SubNetwork/ManagedElement[starts-with(id,"ME2")]

/SubNetwork/ManagedElement[contains(id,"ME")]

The rest of the grammar introduced in the EBNF defines additional rules for the lexical structure (Literal, Number and Name structure). Note also that some functions are excluded since they are not useful for this profile.

#### 4.2.5.4 XPath 3.1

Concepts added in XPath 3.1 [4] provide no obvious value for the use cases analysed.

*Editor's note: Functions to deal with JSON arrays and maps may need further analysis.*

#### 4.2.5.5 JSON Pointer

JSON Pointer [5] defines a string syntax for identifying a specific value within a JSON document. A JSON Pointer expression points to one and only one value. Items of an array are identified based on their index and not key.

These properties do not make JSON Pointer a good candidate for selecting multiple nodes.

#### 4.2.5.6 JSONPath

JSONPath [6] is still a draft RFC. Its intention is to be a powerful JSON Pointer.

**Example JSONPath expressions**

Select a subtree starting from the base object

$

Select all objects with a specific class

$..\*[?(@.objectClass=="ManagedElement")]

Select all objects based on their DN

$..\*[?(@.objectInstance=="DN1" || @.objectInstance== "DN2")]

Select specified attributes and attribute fields:

$.SubNetwork[?(@.id=="SN1")].attributes.userLabel

$.SubNetwork[?(@.id=="SN1")].attributes.plmnId.mcc

$.SubNetwork[?(@.id=="SN1")].attributes[userLabel,userDefinedNetworkType]

$.SubNetwork[?(@.id=="SN1")].attributes[userLabel,plmnId.mcc], does not work

Select an array element

$.SubNetwork[?(@.id=="SN1")].ThresholdMonitor[?(@.id=="TM1")].attributes.thresholdLevels[?(@.level==1)]

Conditional read: return attributes container only when a condition is met,

in this case "userLabel" is equal to "Berlin NW"

$.SubNetwork[?(@.id=="SN1")][?(@.userLabel=="Berlin NW")]

in this case we walk down one level for a test

$.SubNetwork[?(@.id=="SN1")][?(@.plmnId.mcc==456)], does not work

in this case an absolute path is used, that is outside of the current object

$.SubNetwork[?(@.id=="SN1")].attributes[?($.SubNetwork[?(@.id=="SN1")].ThresholdMonitor[?(@.id=="TM1")].attributes.metric=="Metric1")], does not work

Test on the presence of an attribute

$.SubNetwork[?(@.id=="SN1")][?(@.userLabel)]

### 4.2.6 CR proposal

It is proposed to add the following clause to TS 32.158.

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Definitions (to be placed in definitions clause)**

**JSON document**: A JSON document contains JSON text. A JSON text is a serialized JSON value. A JSON value is an object, array, number, or string, or one of the following three literal names: false, null, true.

**X JPath**

**X.1 Introduction**

Information can be represented in a structured way using markup languages. Well-known and widely used markup languages are for example XML and JSON.

It is required for numerous purposes to identify distinct portions of XML or JSON documents. For XML, XPath has been designed for that purpose. XPath is very powerful and includes capabilities for conditional node selection with predicates. XPath expressions can select one or more portions of an XML document.

JSON Pointer serves a similar purpose. However, its capabilities are limited when compared to XPath. JSON Pointer expressions can identify only one portion of a JSON document. The idea of introducing conditions into the information selection process is not supported.

This calls for a notation applicable to JSON documents with more advanced features than JSON Pointer. This notation is called JPath. It is inspired by and based on XPath.

Even though XPath was originally designed to select one or more nodes of an XML document, XPath expressions operate on documents based on a conceptual data model, the XPath data model. A mapping from the XML Information Set to the XPath data model is provided in Annex B of XPath 1.0 [15].

The main purpose of the JPath specification is to provide a mapping from a JSON document to the XPath data model. With this in place XPath expressions are (indirectly) applicable to JSON.

This specification will also introduce a few profiles for XPath. These profiles are designed to provide the functionality required for network and service management.

Clause X.2 provides a short review of the XPath data model. Clause X.3 defines the mapping of a JSON document to the XPath data model, and clause X.4 introduces a few XPath profiles. Annex A demonstrates to use of JPath for network management tasks.

Readers should be familiar with XPath 1.0 [15].

**X.2 XPath data model**

The XPath data model is described in clause 5 of W3C Xpath1.0 specification [15]. It is a conceptual model without formal notation.

The model consists of nodes with relationships between them. There are seven types of nodes defined: root node, element node, text node, attribute node, namespace node, processing instruction node, comment node.

Note that the data model for XPath 2.0 [xx] and XPath 3.1 [xy] is described in XQuery and in XPath Data Model 3.1 [15] and which is out of scope of this specification.

**X.3 Mapping of JSON to the XPath data model**

**X.3.1 Supported JSON documents.**

Only documents whose value is a JSON object are supported. Multiple name/value pairs are allowed.

**X.3.2 Mapping of the JSON document**

A JSON document is mapped to the root node. The root node has no name.

*Editor's note: Discuss mapping of the "top-level" JSON value.*

*Edito's note: Discuss mapping of numbers and strings.*

**X.3.3 Mapping of the name of name/value pairs**

If the value of the name/value pair is a string, a number, one of the tree literal names, or a JSON object, then the name of the name/value pair is mapped to one element node. The name of the element node is equal to the name of the name/value pair.

If the value of the name/value pair is a JSON array, then the name of the name/value pair is mapped to a specific number of element nodes. The names of these element nodes are all identical to the name of the name/value pair. The number of element nodes depends on the data types of the array items and is determined as follows:

- For each array item there is an element node if the array item is a scalar or a JSON object.

- If the array item is an array itself, then there is an element node for each array item of the subordinate array.

**X.3.4 Mapping of the value of name/value pairs**

**X.3.4.1 Mapping of scalar values**

A scalar value is a string, a number, or one of the tree literal names. These values are mapped to text nodes.

A text node coming from the value of a mapped name/value pair is the child of the element node coming from the name of the mapped name/value pair. Vice versa, the element node coming from the name of the mapped name/value pair is the parent of the text node coming from the value of the mapped name/value pair.

Example:

|  |  |
| --- | --- |
| "a": 1 | <a>1</a> |

**X.3.4.2 Mapping of a JSON object**

A JSON object consists of unordered name/value pairs. These name/value pairs are mapped as described in this clause. The element nodes coming from the name/value pairs of a JSON object shall be considered as an unordered list of element nodes.

An element node coming from the name/value pair of a JSON object is a child of the element node coming from the name of the name/value pair, whose value is the JSON object. Vice versa, the element node coming from the name of the name/value pair, whose value is a JSON object, is the parent of the element nodes coming from the name/value pairs of the JSON object.

Example:

|  |  |
| --- | --- |
| "a": {  "b": 1,  "c": 2  } | <a>  <b>1</b>  <c>2</c>  </a> |

**X.3.4.3 Mapping of a JSON array**

A JSON array consists of ordered array items. Each array item can be a string, a number, one of the tree literal names, a JSON object, or a JSON array. The array items are mapped as described in this clause. The element nodes coming from the array items shall be ordered in the same way as the array items.

An element node coming from an array item of a JSON array is a child of the element node coming from the name of the name/value pair, whose value is the JSON array. Vice versa, the element node coming from the name of the name/value pair, whose value is a JSON array, is the parent of the element nodes coming from the array items of the JSON array.

Example:

|  |  |
| --- | --- |
| "a": [  1,  2  ] | <a>1</a>  <a>2</a> |

|  |  |
| --- | --- |
| "a": [  [1, 2],  [3, 4]  ] | <a>1</a>  <a>2</a>  <a>3</a>  <a>4</a> |

|  |  |
| --- | --- |
| "a": [  1,  [2, 3],  {"c": 4}  ] | <a>1</a>  <a>2</a>  <a>3</a>  <a>  <c>4</c>  </a> |

**X.3.5 XPath data model concepts required by JSON**

A JSON document is mapped to root nodes, element nodes and text nodes. Attribute nodes, namespace nodes, processing instruction nodes and comment nodes have no equivalent in JSON.

The concept of document order is applicable only for arrays.

*Editor's note: It is ffs how the concept of document order for arrays can be applied.*

The concept of variables is not used.

**X.3.6 Additional considerations for JSON definitions of a NRM**

When working with the JSON definitions of a NRM the following statements are always true:

- The items of an array have the same type.

- Array items can be only scalars or JSON objects.

*Editor's note: Explain concept of Accessible tree of an NRM*

*Editor's note: Explain that difference between object and attributes disappears in JSON.*

**X.4 JPath expressions**

**X.4.1 Introduction**

JPath uses the same syntax, the same concepts and the same definitions as XPath. JPath expressions are a subset of XPath expressions. All subsets support only the abbreviated syntax. The output of a JPath expression is always a node set.

Different subsets are defined in the following clauses. All subsets have the same allowed location path expression. Subsets differ in the capabilities of the predicate. A subset is also called JPath profile.

**X.4.1 Evaluation context**

JPath expressions are evaluated in a context, that is a subset of the XPath evaluation context. The JPath context includes

- a node (the context node)

- a pair of non-zero positive integers (the context position and the context size)

- a function library

**X.4.3 The location path**

A JPath expression is an absolute location path. An absolute location path consists of "/", optionally followed by a relative location path. A "/" by itself selects the root node of the document.

AbsoluteLocationPath ::= '/' RelativeLocationPath

A relative location path consists of a sequence of one or more location steps separated by "/".

RelativeLocationPath ::= Step | RelativeLocationPath '/' Step

Only the child axis is supported. The child axis is the default axis and omitted in the abbreviated syntax of a location step. The location step contains only a node test and an optional predicate.

Step ::= NodeTest Predicate?

The node test is a name test. The asterisk "\*" is supported and selects all element children of the context node.

NodeTest ::= NameTest

NameTest ::= '\*' | QName

The "QName" is either a class name, the string "attributes", an attribute name, or an attribute field name.

*Editor's note: EBNF for QName tbd.*

The predicate is an expression encapsulated in rectangular brackets.

Predicate ::= '[' PredicateExpr ']'

The capabilities of the predicate expression differ for the different JPath profiles.

*Editor's note: Add XPath 2.0 capability to select multiple nodes with a sequence, e.g.* "*…/(a,b)*".

**X.4.4 JPath basic**

tbd

**X.4.5 JPath advanced**

tbd

### 4.2.7 Conclusion

It is recommended to define a mapping from a JSON document to the XPath data model and to identify a basic and advanced subset of XPath expressions working on that model.

This allows to apply a certain subset of XPath expressions safely to JSON documents.

It is also recommended to use the mechanism not only for targeted notification subscriptions, but also whenever it is required to identify, when being located on some object or attribute, other attributes in other objects.

## 4.3 Issue #3: Definition of createMOI

### 4.3.1 Issue description

The operation "createMOI" in TS 28.532 is underspecified and needs to be improved in many aspects.

### 4.3.2 Analysis

The following points need to be clarified in the definition of "createMOI":

- Clarify that the name of the new object is assigned by the MnS consumer.

- Clarify that the MnS consumer does not need to provide in the creation request values for all attributes defined for the object class.

- Clarify that depending on the object class some attribute values have to be provided in the creation request.

- Clarify the MnS producer may provide values for attributes, for which no value is provided in the request.

- Clarify the operation is synchronous.

### 4.3.3 CR proposal

The existing clause 11.1.1.1 in TS 28.532 shall be replaced as follows:

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11.1.1.1 createMOI operation

11.1.1.1.1 Description

This operation is invoked by MnS consumers to request a MnS producer to create a (single) managed object instance on the MnS producer.

The "managedObjectClass" parameter in the request specifies the class name and the "managedObjectInstance" parameter the instance name of the object to be created. Both parameters shall be included in the request.

The MnS consumer shall generate the instance name by first assigning a value to the naming attribute of the new instance, and then constructing a DN according to TS 32.300 [25].

The MnS consumer shall provide in "attributeListIn" none, some or all values for the attributes specified by the managed object class definition of the class to be created. The MnS producer shall not update attribute values or remove attribute values, that are provided in the request, before creating the object and returning the "createMOI" response.

The properties of an attribute determine if attribute values shall, shall not or may be provided in the "createMOI" request. If no value is provided and a default value is specified for the attribute, the MnS producer shall set the attribute value to the default value. For further information on attribute properties and their impact on the presence or absence of attribute values in object creation requests and MnS producer behaviour, see TS 32.156 [x], clause 5.2.1 and annex B.

When the MnS producer assigns values, that are not known to the MnS consumer, to one or more attributes for which no value is included in the "createMOI" request, the MnS producer shall include "attributeListOut" in the "createMOI" response, otherwise "attributeListOut" may be omitted.

In case of a successful operation, the object shall be created immediately upon reception of the "createMOI" request, and the "createMOI" response shall be returned immediately after the creation of the object. The MnS producer shall not wait with the creation of the object or returning the response until some other potentially long-lasting process or activity, that might be triggered by the reception of the request or the creation of the object, has completed.

Only objects, whose parent exists, can be created (directly under that parent). The MnS producer shall consider an attempt to create an object whose parent object does not exist as an error.

Note, stage 3 protocols represent missing values for attributes, that are defined by the object class, in the "createMOI" request in different ways. For some protocols just the attribute name may be present, without an attribute value. For other protocols, the complete attribute name/value pair may be absent.

Some stage 3 protocols do not support returning "attributeListOut". In this case, the MnS producer shall not modify the attribute list provided in the request before creating the object. If required by TS 32.156 [x], clause 5.2.1 and annex B, the MnS producer shall assign default values to attributes only after returning the "createMOI" response. Attribute value change notifications may be used to notify MnS consumers about the changes. Only default values, that have a specific definitive value may be assigned upon object creation. This is because the MnS consumer knows the MnS producer will assign this value when required according to TS 32.156 [x], clause 5.2.1 and annex B. Default values that are determined by the MnS producer based on standardized or proprietary selection methods are typically not known to MnS consumers.

11.1.1.1.2 Input parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | S | Information Type / Legal Values | Comment |
| managedObjectClass | M | string | Class name of the managed object to be created. |
| managedObjectInstance | M | DN | Instance name of the managed object to be created. |
| attributeListIn | M | LIST OF SEQUENCE< attribute name, attribute value> | List of attribute name/value pairs of the managed object to be created. |

11.1.1.1.3 Output parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | S | Matching Information / Legal Values | Comment |
| attributeListOut | O | LIST OF SEQUENCE< attribute name, attribute value> | List of attribute name/value pairs of the created object.  The parameter shall be present when the MnS producer assigns values, that are not known to the MnS consumer, to one or more attributes, otherwise it may be absent. |
| Status | M | ENUM (OperationSucceeded, OperationFailed) |  |

\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 4.3.4 Conclusion

It is recommended to update the definition of the operation "createMOI" in clause 11.1.1.1 of TS 28.532 as defined in clause 4.3.3 of the present document.

## 4.4 Issue #4: Definition of modifyMOIAttributes

### 4.4.1 Issue description

The operation "modifyMOIAttributes" in TS 28.532 is underspecified and needs to be improved in many aspects.

### 4.4.2 Analysis

### 4.4.3 CR proposal

As a solution for the issue, the existing clause 11.1.1.3 in TS 28.532 could be replaced as follows:

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11.1.1.3 modifyMOIAttributes operation

11.1.1.3.1 Definition

This operation is invoked by MnS consumers to request a MnS producer to modify one or more attributes of one or more manged objects on that MnS producer.

The type of modifications, that can be performed, depend on the attribute multiplicity.

With the "multiplicity: 1", the following modifications are possible:

- Replacement of an attribute value or attribute field value.

With the "multiplicity: 0..1", the following modifications are possible:

- Replacement of an attribute value or attribute field value, applicable to attributes or attribute fields that do have a value when performing the operation.

- Addition of an attribute value or attribute field value, applicable to attributes or attribute fields that do not have a value when performing the operation.

- Removal of an attribute value or attribute field value, applicable to attributes or attribute fields that do have a value when performing the operation.

With the "multiplicity: 1..n", the following modifications are possible:

- Replacement of an existing attribute element or existing attribute field element.

- Addition of a new attribute element or attribute field element.

- Removal of an attribute element or attribute field element, applicable to attribute or attribute fields that do have at least two elements.

With the "multiplicity: 0..n", the following modifications are possible:

- Replacement of an existing attribute element or existing attribute field element.

- Addition of an attribute element or attribute field element.

- Removal of an attribute element or attribute field element, applicable to attribute or attribute fields that do have at least one element.

The selection of the objects to be modified is achieved with the parameters "baseObjectInstance", "scopeType", "scopeLevel" and "filter". If no "scopeType" is specified, the value defaults to "BASE\_ONLY". Also, when no scoping is supported by the MnS producer, the value in "baseObjectInstance" identifies the object to be modified.

A protocol may choose to split the selection of objects with scoping and filtering and the modification of the attributes of the selected objects into different operations.

The modifications to be applied to the selected objects are described in the "modifications" parameter. This is a multi-valued parameter. Each value is a structure composed of the "modifyOperator", the "nodeIdentifier" and the "nodeValue". The values of "modificationList" are ordered and shall be applied in the sequence as they occur in the list.

The parameter "nodeIdentifier" is used to identify the attributes or attribute fields to which modifications shall be applied. For simple attributes this can be just the attribute name. For structured attributes, however, attribute fields need to be identified. Different attribute fields pertaining to the same attribute may have the same name. For that reason, unambiguous identification of attribute fields is not possible with their names only. The identification needs to be based e.g. on global or local identifiers, or on the specification of a path allowing to navigate to the attribute field. Details are protocol specific.

For the modification of multi-valued attributes or multi-valued attribute fields two cases are distinguished:

- All values (elements) are considered as a single value. Elements cannot be addressed individually. If replaced, all existing elements are replaced with the new elements received in the modification request. If deleted, all existing elements are deleted. Addition of new elements to existing elements is not possible.

- Each element can be addressed individually. Single elements can be added, replaced, and deleted.

The first case does not need any further considerations. The second case requires the identification of attribute elements and attribute field elements in the modification request, when replacing and deleting elements in ordered and unordered lists, and when adding elements to ordered lists. Details are not defined at stage 2. They are protocol specific and include identification by the element value, identification by a positional index or identification by an auxiliary key added at stage 3. Note that the concept of element keys is not defined at stage 2. Identification of elements may be provided also by the "nodeIdentifier" parameter.

The "modifyOperator" parameter specifies the modification to be applied to the value of the attribute or attribute field, or the attribute element or attribute field element identified by the "nodeIdentifier". The parameter can have the values "replace", "add", "remove" or "setToDefault":

- For "replace", the "nodeValue" specifies the attribute value, attribute field value, attribute element or attribute field element that shall replace the existing value.

- For "add", the "nodeValue" specifies the attribute value or attribute field value to be added to an attribute or attribute field without value, or the new attribute element or attribute field element to be added to a multi-valued attribute.

- For "remove", the "nodeValue" is absent when an attribute value or attribute field value is removed. When an attribute element or attribute field element is removed, "nodeValue" may carry the element to be removed, depending on how on protocol level attribute elements and attribute field elements are identified.

- For "setToDefault", the "nodeValue" is absent.

Attributes and attribute fields without value can be represented in different protocol specific ways, for example by an attribute name without attribute value, by an absent attribute name/value pair, or by a specific attribute value (such as "null" or "nil").

The "modifyMOIAttributes" operation allows to modify one or more attributes in one or more objects. When not all attribute modifications can be applied successfully, the MnS producer has different options how to proceed. He may not perform any of the modifications and roll back to the state at the reception of the modification request. He may apply the changes that can be applied, so that some of the requested modifications are applied and some are not applied. He may stop processing the modification request when the first error occurs. The stage 2 definition of this operation does not include any provisions on how to proceed in case an error occurs. These provisions are left to stage 3.

When attribute properties as defined in clause 5.2.1 of TS 32.156 [?] are used, then a MnS consumer should take into account the properties "type", "allowedValues", "multiplicity", "isOrdered", "isUnique", "isNullable" and "passedById" when constructing the attribute modification request. If ignored, the MnS producer shall reject the request and generate an error response.

Furthermore, when attribute properties are used, then a MnS consumer can modify an attribute with "modifyMOIAttributes" only if "isInvariant: False" and "isWritble: True". Any permission to modify attributes as determined by these attribute properties may be overridden by access control.

11.1.1.3.2 Input parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | S | Matching Information / Legal Values | Comment |
| baseObjectInstance | M | ManagedEntity.objectInstance | Base object used for scoping the target objects of the operation. If no scoping is applied, the base object is the only target object. |
| scopeType | O | See corresponding parameter in "getMOIAttributes". | See corresponding parameter in "getMOIAttributes". |
| scopeLevel | O | See corresponding parameter in "getMOIAttributes". | See corresponding parameter in "getMOIAttributes". |
| filter | O | See corresponding parameter in "getMOIAttributes". | See corresponding parameter in "getMOIAttributes". |
| modificationsIn | M | LIST OF SEQUENCE <  nodeIdentifier  modifyOperator,  nodeValue  > | Set of sub-operations to be applied to attributes and attribute fields of the target objects  The "nodeIdentifier" specifies the target attribute or target attribute field of the sub-operation.  The "modifyOperator" specifies the operation to be applied to the target attribute or target attribute field . The parameter can have the values "replace", "add", "remove" or "setToDefault".  The "nodeValue" specifies the value used by the sub-operation. This parameter is absent for "remove" operations. |

11.1.1.3.3 Output parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | S | Matching Information / Legal Values | Comment |
| modificationsOut | M | LIST OF SEQUENCE <  objectInstance DN,  objectClass string,  LIST OF SEQUENCE<  attribute name,  attribute value >  > | Provides for each object, that is selected by the request, the object name, the object class, and a list of name/value pairs with the values of *all* attributes after modification.  If all requested modifications are applied, the parameter may be absent.  If no requested modification is applied and an error response is returned, the parameter may be absent, too.. |
| status | M | ENUM (  SUCCEEDED,  PARTIALLY\_FAILED,  FAILED  ) | Indicates if all, some or none of the requested modifications were applied. Details on the error, such as which modification could not be applied and the corresponding reason, may be returned as well. |

\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 4.4.4 Conclusion

It is recommended to update the definition of the operation "modifyMOIAttributes" in clause 11.1.1.3 of TS 28.532 as defined in clause 4.4.3 of the present document.

## 4.5 Issue #5: HTTP error response format

### 4.5.1 Issue description

When a MnS producer cannot fulfil a HTTP request, the MnS producer should provide in the error response detailed information about what exactly failed and for what reason. Ideally the error response body returned is standardized.

### 4.5.2 Analysis

The error response of CRUD operations is currently defined as

ErrorResponse:

description: >-

Default schema for the response message body in case the request

is not successful.

type: object

properties:

error:

type: object

properties:

errorInfo:

type: string

The error response is a JSON object with a single property "errorInfo" that is of type string. Vendors can provide error information in "errorInfo" and extend the object with addition properties. The standard provides hence just a container for vendor specific error information. Error details cannot be provided in a standardized way. An error response format should be standardized.

IETF RFC 7807 [12] provides an example how the issue has been approached in the industry.

### 4.5.3 CR proposal

The following clause shall be added to TS 32.158.

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

X Error responses

*Editor's note: Interaction with access control is to be studied.*

X.1 HTTP error codes

In case of an error on the MnS Producer an error response status code of the classes 4xx (Client Error) or 5xx (Server Error) is returned to the MnS Consumer in the response status line. A complete list of error status codes is maintained by IANA.

Table X.1-1 and Table X.1-2 list the status codes that shall be supported by MnS Producer and MnS Consumer implementations compliant to this specification.

Table X.1-1: Supported 4xx client error status codes

|  |  |  |
| --- | --- | --- |
| Error status code | Reference | Description |
| 400 Bad Request | IETF RFC 7231 [2] | indicates that the server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing). |
| 403 Forbidden | IETF RFC 7231 [2] | indicates that the server understood the request but refuses to authorize it. |
| 404 Not Found | IETF RFC 7231 [2] | indicates that the origin server did not find a current representation for the target resource or is not willing to disclose that one exists. |
| 405 Method Not Allowed | IETF RFC 7231 [2] | indicates that the method received in the request-line is known by the origin server but not supported by the target resource. |
| 406 Not Acceptable | IETF RFC 7231 [2] | indicates that the target resource does not have a current representation that would be acceptable to the user agent, according to the proactive negotiation header fields received in the request (Section 5.3), and the server is unwilling to supply a default representation. |
| 408 Request Timeout | IETF RFC 7231 [2] | indicates that the server did not receive a complete request message within the time that it was prepared to wait. |
| 410 Gone | IETF RFC 7231 [2] | indicates that access to the target resource is no longer available at the origin server and that this condition is likely to be permanent. |
| 411 Length Required | IETF RFC 7231 [2] | indicates that the server refuses to accept the request without a defined Content-Length field containing the length of the message body in the request message. |
| 413 Payload Too Large | IETF RFC 7231 [?2 | indicates that 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 [2] | indicates that the server is refusing to service the request because the request-target is longer than the server is willing to interpret. |
| 415 Unsupported Media Type | IETF RFC 7231 [2] | indicates that the origin server is refusing to service the request because the payload is in a format not supported by this method on the target resource. |
| 422 Unprocessable Entity | IETF RFC 4918 [?] | indicates the server understands the content type of the request entity (hence a 415(Unsupported Media Type) status code is inappropriate), and the syntax of the request entity is correct (thus a 400 (Bad Request) status code is inappropriate) but was unable to process the contained instructions. |
| 426 Upgrade Required | IETF RFC 7231 [2] | indicates that the server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol. |
| 429 Too Many Requests | IETF RFC 6585 [?] | indicates that the user has sent too many requests in a given amount of time ("rate limiting"). |

Table X.1-2: Supported 5xx server error status codes

|  |  |  |
| --- | --- | --- |
| Error status code | Reference | Description |
| 500 Internal Server Error | IETF RFC 7231 [2] | Indicates that the server encountered an unexpected condition that prevented it from fulfilling the request. |
| 501 Not Implemented | IETF RFC 7231 [2] | indicates that the server does not support the functionality required to fulfill the request. |
| 502 Bad Gateway | IETF RFC 7231 [2] | indicates that the server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request. |
| 503 Service Unavailable | IETF RFC 7231 [2] | indicates that the server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay. |
| 504 Gateway Timeout | IETF RFC 7231 [2] | indicates that the server, while acting as a gateway or proxy, did not receive a timely response from an upstream server it needed to access in order to complete the request. |
| 505 HTTP Version Not Supported | IETF RFC 7231 [2] | indicates that the server does not support, or refuses to support, the major version of HTTP that was used in the request message. |

A MnS Producer may use other error response codes as well. However, there is no guarantee that a MnS Consumer understands the semantics beyond what is specified in clause 6 of IETF RFC 7231 [2]: "A client MUST understand the class of any status code, as indicated by the first digit, and treat an unrecognized status code as being equivalent to the x00 status code of that class".

X.2 Error response body

X.2.1 Overview

HTTP status codes provide high level error information. This is often not sufficient, for example in situations where the MnS Producer wants to convey error information to aid the MnS Consumer in generating a valid request. In these cases, the MnS Producer needs to include an error response body in the response.

The error response body specified in the present document is derived from IETF RFC 7807 [??], clause 3.1. The following definitions are re-used and shall be supported:

- The "status" property contains the status code for the error..

- The "type" property provides high level error information.

- The "title" provides a short, human-readable summary of the problem type. It shall not change from occurrence to occurrence of the problem.

The "details" and "instance" properties defined in IETF RFC 7807 [??] are not re-used. Potential support is outside of the provisions of the present document.

The three re-used properties are extended with the following property that shall be supported.

- The "reason" (string) property" provides more details on the error conditions than "type".

The "status", "type", "title" and "reason" property are applicable to all HTTP methods and request media types. In addition, the following method specific properties are defined:

- The "badQueryParams" property provides information about bad query parameters in GET requests.

- The "badOp" property specifies the operation in JSON Patch and 3GPP JSON Patch requests, that cannot be satisfied.

- The "badAttributes" property provides information about bad attributes in PUT, POST, JSON Merge Patch and 3GPP JSON Merge Patch requests.

- The "badObjects" property provides information about bad objects in 3GPP JSON Merge Patch requests.

An error response provides details on one or more problems in the request. Each problem is described by a subset of the seven properties specified above. Each problem is related to a specific problem type that is identified by the value of the "type" property.

A single request may have more than one problem. This situation may occur for example when a GET request has multiple query parameters, or when a PATCH request contains multiple operations to be applied to the information model on the MnS Producer.

The MnS Producer shall include at least details on one problem in the error response. A MnS Consumer cannot assume that the returned list of problems is exhaustive and includes all problems in the request. A MnS producer may stop processing the request upon detection of the first problem and return an error response.

If all problems have the same error status code, that code shall be used in the status line of the error response. The "status" property of each problem description may repeat that code. However, if the problems have different error codes, the "207 Multi-Status" (IETF RFC4918 [???]) code shall be used in the response status line. The "status" property related to each problem shall contain the specific status code.

The concrete format of the error response body depends on the request. The media type for all error response formats is "application/vnd.3gpp.error+json". The following clauses provide the details.

X.2.2 Error response format for GET requests

For a successful GET request all query parameters in the request need to be understood, accepted and satisfied by the MnS Producer. In case one or more bad query parameters are present in the request an error response shall be returned.

The error response is a JSON array of JSON objects with "status", "type", "title", "reason", and "badQueryParams" properties. The "type" property shall be present. The "status" property shall be present under the conditions specifed in clause X.2.1. The "title", "reason" and "badQueryParams" properties may be present.

The MnS Producer shall include at least one bad query parameter in the "badQueryParams" property. A MnS Consumer cannot assume that the returned list of bad query parameters includes all bad parameters in the request. A MnS producer may stop processing the request upon detection of the first bad query parameter and return an error response.

|  |
| --- |
| {  "type": "array",  "items": {  "type": "object",  "properties": {  "status": {"type": "string"},  "type": {"type": "string"},  "reason": {"type": "string"},  "title": {"type": "string"},  "badQueryParams": {"type": "array","items": {"type": "string"}}  },  "required": ["type"]  }  } |

X.2.3 Error response format for PUT, POST and DELETE requests

The error response is a JSON array of JSON objects with "status", "type", "title", and "reason" properties. The "type" property shall be present. The "status", "title" and "reason" properties may be present.

|  |
| --- |
| {  "type": "array",  "items": {  "type": "object",  "properties": {  "status": {"type": "string"},  "type": {"type": "string"},  "reason": {"type": "string"},  "title": {"type": "string"}  },  "required": ["type"]  }  } |

X.2.4 Error response format for JSON Patch and 3GPP JSON Patch requests

For a successful JSON Patch or 3GPP JSON Patch request all operations in the patch request need to be understood, accepted, and satisfied by the MnS Producer. In case one or more operations in the request cannot be applied, no operation shall be applied, and an error response shall be returned. JSON Patch and 3GPP JSON Patch are atomic, as described in clause 6.3.1.

The error response is a JSON array of JSON objects with "badOp", "status", "type", "title" and "reason" properties. The "badOp" and "type" properties shall be present. The "status" property shall be present only under the conditions specified in clause X.2.1.

Each object provides information about one bad patch operation. The patch operation, that cannot be satisfied, is identified with "badOp", whose value is a JSON Pointer identifying the object with the bad patch operation in the request. Successful patch operations are not included in the response. The order of the operations in the response shall be the same as in the request.

The MnS Producer shall include at least one bad operation in the error response. A MnS Consumer cannot assume that the returned list of bad operations includes all bad operations in the request. A MnS Producer may stop processing the request upon detection of the first bad operation and return an error response.

|  |
| --- |
| {  "type": "array",  "items": {  "type": "object",  "properties": {  "badOp": {"type": "string"},  "status": {"type": "string"},  "type": {"type": "string"},  "reason": {"type": "string"},  "title": {"type": "string"}  },  "required": ["badOp", "type"]  }  } |

X.2.5 Error response format for JSON and 3GPP JSON Merge Patch requests

Like JSON Patch and 3GPP JSON Patch, JSON and 3GPP JSON Merge Patch are atomic.

The error response is a JSON array of JSON objects with "status", "type", "title", "reason", "badAttributes", and "badObjects" properties. The "type" property shall be present. The "status" property shall be present only under the conditions specified in clause X.2.1. The "badObjects" property is applicable only for 3GPP JSON Merge Patch.

The path to the bad attributes or bad objects is a relative URI and constructed according to the rules defined in clause 6.4.3 for the "path" property of 3GPP JSON Patch.

The MnS Producer shall include at least one bad attribute in the "badAttributes" property, or one bad object in the "badObjects" property. A MnS Consumer cannot assume that the returned list of bad attributes or objects includes all bad attributes or objects in the request. A MnS producer may stop processing the request upon detection of the first bad attribute or object and return an error response.

|  |
| --- |
| {  "type": "array",  "items": {  "type": "object",  "properties": {  "status": {"type": "string"},  "type": {"type": "string"},  "reason": {"type": "string"},  "title": {"type": "string"},  "badAttributes": {"type": "array","items": {"type": "string"}},  "badObjects": {"type": "array","items": {"type": "string"}},  },  "required": ["type"]  }  } |

X.3 The "type" property

The "type" property provides high level error information allowing to complement HTTP 4xx and 5xx error codes in case this is necessary or desired. It provides more details on the nature of the problem than the HTTP error codes. Problem types are specified for the following error response codes.

- 400 Bad Request

- 403 Forbidden

- 422 Unprocessable Content

- 500 Internal Server Error

- 503 Service Unavailable

Note that some error codes convey already all information that can be conveyed. For example, the 404 Not Found status code indicates that the target resource does not exist or has no current representation. It is hard to see which information should be added to make the error response more helpful for the MnS Consumer.

The "type" property is an enumeration of string values. A MnS Producer should use the following values in error situations. Other values may be used as well if deemed more appropriate for specific errors.

- VALIDATION\_ERROR (HTTP error code: 400 Bad Request): The request message does not validate and cannot be processed. Validation refers to two aspects: Validation of the received request message against the JSON schema definition of the request message, and validation of the information model state after applying the requested changes against the JSON schema definition of the information model, for example, if a new instance of a certain object class is allowed to be contained under the class of the specified parent object. The "reason" property may be present and specify more detailed error information.

- REQUEST\_OBJECT\_TREE\_MISMATCH (HTTP error code: 422 Unprocessable Entity): The request message is well formed and understood but cannot be completed due to the current state of the object tree on the MnS Producer. For example, this reason is used when an object is requested to be created below a parent object that does not exist. The "reason" property may be present and specify more detailed error information.

- IE\_NOT\_FOUND (related error code: 400 Bad Request): The information element (object, attribute, attribute field, attribute element) requested to be modified does not exist. The "reason" property may specify more detailed error information. The "reason" property may be present and specify more detailed error information.

- MODIFICATION\_NOT\_ALLOWED (HTTP error code: 403 Forbidden): The requested modification is understood but not allowed. The "reason" property may be present and specify more detailed error information.

- RETRIEVAL\_NOT\_ALLOWED (HTTP error code: 403 Forbidden): The retrieval of the specified information is not allowed. The "reason" property may be present and specify more detailed error information.

- SERVER\_LIMITATION (HTTP error code: 500 Internal Server Error): The request is correct and understood by the MnS Producer, but the MnS Producer cannot satisfy the request due to server limitations. The "reason" property may be present and specify more detailed error information.

- SERVICE\_DISABLED (HTTP error code: 503 Service Unavailable): The MnS Producer has disabled itself and is currently unable or unwilling to handle the request. This condition may occur, for example, in overload situations. The "reason" property shall be absent.

- APPLICATION\_LAYER\_ERROR (HTTP error code: 500 Internal Server Error): The request is correct and understood by the MnS Producer, but the MnS Producer cannot satisfy the request due to application layer issues. The "reason" property may be present and specify more detailed error information.

X.4 The "reason" property

X.4.1 Overview

The "reason" property provides more details on the error conditions than the "type" property. For client-side errors, these reasons may provide hints to the MnS Consumer on how to generate a request without errors. For server-side errors, they may help the MnS Consumer to generate requests that may be satisfied by the MnS Producer.

When multiple reasons apply, the most fundamental reason shall be put in the "reasons" property. For example, when a MnS Consumer attempts to replace an invariant attribute, and - in addition - the attribute value is invalid, then only the information that the attribute is invariant shall be contained in the "reason" property.

The "reason" property may be omitted when the MnS Producer does not want to disclose details on the error to the MnS Consumer.

Detailed error reasons are specified by the "reason" property for the following error codes:

- 400 Bad Request

- 403 Forbidden

- 422 Unprocessable Entity

- 500 Internal Server Error

Error reasons depend on the HTTP method and on if attributes or objects are manipulated. The following clauses specify the details.

X.4.2 Error reasons for GETValid values for the "reason" property for an error response related to HTTP GET are:

- RESPONSE\_TOO\_LARGE (related type: SERVER\_LIMITATION, 500 Internal Server Error): The content requested to be returned exceeds the response body size limit of the MnS Producer.

- NO\_DATA\_ACCESS (related type: SERVER\_LIMITATION, 500 Internal Server Error): The request is correct and understood by the MnS Producer, but the MnS Producer cannot access the requested data.

- QUERY\_MALFORMED (related type: VALIDATION\_ERROR, 400 Bad Request): The syntax of the query component is malformed. The "badQueryParams" property shall be absent.

- QUERY\_PARAM\_NAMES\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more query parameter names are invalid. The "badQueryParams" property shall indicate the names of the invalid parameters.

- QUERY\_PARAM\_VALUES\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more query parameters have an invalid value. The "badQueryParams" property shall indicate the names of the parameters with invalid value.

- QUERY\_PARAMS\_MISSING (related type: VALIDATION\_ERROR, 400 Bad Request): One or more query parameters, that shall be present in the request or that shall be present in case another parameter is present, are missing in the query component. The "badQueryParams" property shall indicate the names of the missing parameters.

- QUERY\_PARAMS\_INCONSISTENT (related type: VALIDATION\_ERROR, 400 Bad Request): Query parameters with mutual dependency constraints do not respect these constraints. The "badQueryParams" property shall indicate the names of the parameters not respecting the dependency constraints.

- ATTRIBUTES\_NOT\_READABLE (related type: RETRIEVAL\_NOT\_ALLOWED ,403 Forbidden): One or more attributes or attribute fields identified by the query parameters are not readable, according to the attribute property "isReadable". The "badQueryParams" property shall indicate the names of the parameters identifying attributes that are not readable.

- QUERY\_PARAMS\_TOO\_COMPLEX (related type: SERVER\_LIMITATION, 500 Internal Server Error): The query parameters and their values are valid but one or more of them cannot be processed as requested because complexity limits of the MnS Producer are reached, for example, a filter expression is syntactically correct but cannot be evaluated and yields no results since the expression is longer or more complex than the MnS producer can or is willing to process. The "badQueryParams" property shall indicate the names of the parameters that cannot be processed.

The list of valid values is not exhaustive. Other values may be used as well. The name style of these enumeration literals shall follow clause 5.3.5.3 of TS 32.156 [?].

Note that is not an error when query parameters do not identify any resource or attribute to be returned.

Note that the following query parameters are currently specified in the present document: "scopeType", "scopeLevel", "filter", "attributes", and "fields".

**Example:**

For the following request

|  |
| --- |
| GET /SubNetwork=SN1?scopeType=COMPLETE\_SUBTREE&scopeLevel=HIGHEST&\  attributeFields=userLabel HTTP/1.1  Host: example.org  Accept: application/json |

the error response may look like

|  |
| --- |
| HTTP/1.1 400 Bad Request  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "VALIDATION\_ERROR",  "reason": "QUERY\_PARAM\_VALUES\_INVALID"  "title": "The value of one or more query parameters is invalid.",  "badQueryParams": ["scopeType", "scopeLevel"]  },  {  "type": "VALIDATION\_ERROR",  "reason": "QUERY\_PARAM\_VALUES\_INVALID",  "title": "The name of one or more query parameters is invalid.",  "badQueryParams": ["attributeFields"]  }  ] |

When the sub-errors have a different HTTP error code, "207 Multi-Status" is used in the response status line. The "status" property is present for each sub-error and specifies the status code for this sub-error.

|  |
| --- |
| HTTP/1.1 207 Multi-Status  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "status": 404,  "type": "VALIDATION\_ERROR",  "reason": "QUERY\_PARAM\_VALUES\_INVALID",  "title": "The value of one or more query parameters is invalid.",  "badQueryParams": ["attributes", "fields"]  },  {  "status": 404,  "type": "VALIDATION\_ERROR",  "reason": "QUERY\_PARAM\_NAMES\_INVALID",  "title": "The name of one or more query parameters is unknown.",  "badQueryParams": ["attributeFields"]  },  {  "status": 500,  "type": "SERVER\_LIMITATION",  "reason": "QUERY\_PARAMS\_TOO\_COMPLEX",  "title": "The semantics of one or more query parameters is too complex to be processed.",  "badQueryParams": ["fields"]  }  ] |

X.4.3 Error reasons for attribute manipulations

X.4.3.1 JSON Patch and 3GPP JSON Patch

This clause specifies reasons for errors that may occur when attempting to manipulate attributes of existing resources with JSON Patch and 3GPP JSON Patch. JSON Patch and 3GPP JSON Patch are used for partial resource updates.

This specification defines the following error reasons for use with JSON Patch and 3GPP JSON Patch:

- NEW\_ATTRIBUTE\_VALUE\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): The attribute, attribute field or attribute element, as specified in the "path" property, cannot be added, or its value cannot be replaced, as requested, because the value, as specified in the "value" property, is invalid. Valid values are determined by the attribute properties "type", "allowedValues", "multiplicity", "isOrdered", "isUnique", and "isNullable".

- NEW\_ATTRIBUTE\_NAME\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): The attribute or attribute field cannot be added as requested, because its name, as specified in the "path" property, is invalid.

- NEW\_ATTRIBUTE\_PARENT\_NOT\_FOUND (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Entity): The attribute or attribute field cannot be added as requested, because its parent, as specified in the "path" property, does not exist.

- ATTRIBUTE\_NOT\_FOUND (related type: IE\_NOT\_FOUND, 400 Bad Request): The attribute or attribute field cannot be removed, moved, copied, or is value cannot be replaced, as requested, because the "path" or "from" property identifies an attribute or attribute field, that does not exist.

- ATTRIBUTE\_ELEMENT\_NOT\_FOUND (related type: IE\_NOT\_FOUND, 400 Bad Request): The attribute element cannot be replaced, removed, moved, or copied, because the "path" or "from" property identifies an attribute element, that does not exist.

- ATTRIBUTE\_INDEX\_BAD (related type: IE\_NOT\_FOUND, 400 Bad Request): The attribute element cannot be added at the specified array location as requested, because the array element index specified in the "path" property is greater than the number of elements in the array.

- FINAL\_MV\_ATTRIBUTE\_VALUE\_INVALID (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Entity): The attribute element, as specified in the "value" property cannot be added to or removed from the multi-valued attribute as requested, because this would result in an invalid value, according to the attribute properties "multiplicity" or "isUnique". The attribute element itself is valid.

- ATTRIBUTE\_NOT\_WRITABLE (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): The attribute or attribute field cannot be added, removed, or moved, or its value cannot be replaced, as requested, because the attribute or attribute field is not writable by MnS Consumers, according to the attribute property "isWritable".

- ATTRIBUTE\_INVARIANT (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): The attribute or attribute field cannot be added, removed, or moved, or its value cannot be replaced, as requested, because the attribute or attribute field is invariant, according to the attribute property "isInvariant".

- OP\_UNKNOWN (related type: VALIDATION\_ERROR, 400 Bad Request): The patch operation specified by the "op" property is not known by the MnS producer. This situation may occur, for example, when a patch operation is not supported or wrongly spelled.

The list of error reasons is not exhaustive. Other values may be defined and used as well. The name style of these enumeration literals shall follow clause 5.3.5.3 of TS 32.156 [?].

X.4.3.2 JSON Merge Patch, 3GPP JSON Merge Patch and PUT

This clause specifies reasons for errors that may occur when attempting to manipulate attributes of existing resources with JSON Merge Patch, 3GPP JSON Merge Patch and PUT. JSON Merge Patch and 3GPP Merge JSON Patch are used for partial resource updates. PUT is used for complete resource updates.

The following error reasons are defined for use with JSON Merge Patch, 3GPP JSON Merge Patch, and PUT:

- NEW\_ATTRIBUTE\_VALUE\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more attributes or attribute fields cannot be added, or their values cannot be replaced, as requested, because the received value is invalid. Valid values are determined by the attribute properties "type", "allowedValues", "multiplicity", "isOrdered", "isUnique", and "isNullable". The "badAttributes" property provides the path to these attributes and attribute fields.

- NEW\_ATTRIBUTE\_NAME\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more attributes or attribute fields cannot be added as requested, because the received attribute or attribute field name is invalid. The "badAttributes" property provides the path to these attributes and attribute fields.

- ATTRIBUTE\_NOT\_WRITABLE (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): One or more attributes or attribute fields cannot be added or removed, or their values cannot be replaced, as requested, because the attributes or attribute fields are not writable by MnS Consumers, according to the attribute property "isWritable". The "badAttributes" property provides the path to these attributes and attribute fields.

- ATTRIBUTE\_INVARIANT (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): One or more attributes or attribute fields cannot be added or removed, or their values cannot be replaced, as requested, because the attributes or attribute fields are invariant, according to the attribute property "isInvariant". The "badAttributes" property provides the path to these attributes and attribute fields.

The following additional error reasons are defined for use with JSON Merge Patch and 3GPP JSON Merge Patch:

- ATTRIBUTE\_NOT\_FOUND (related type: IE\_NOT\_FOUND, 400 Bad Request): One or more attribute or attribute fields cannot be removed as requested, because they do not exist. The "badAttributes" property provides the path to these attributes and attribute fields.

The list of error reasons is not exhaustive. Other values may be defined and used as well. The name style of these enumeration literals shall follow clause 5.3.5.3 of TS 32.156 [?].

X.4.3.3 Examples

**JSON Patch example 1 ("add"):**

In the first example an attribute field is requested to be added.

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

When "attrA" is invariant, the error response might look like.

|  |
| --- |
| HTTP/1.1 403 Forbidden  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_INVARIANT",  "title": "The attribute field, whose value is requested to be replaced, is invariant."  }  ] |

Other responses may look as below. Note that information not essential for illustrating the examples is left out (referred to as "...")

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "NEW\_ATTRIBUTE\_NAME\_INVALID",  "title": "The name of the attribute, that is requested to be added, is invalid."  }  ] |

when "attrB" is not known or supported.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "NEW\_ATTRIBUTE\_VALUE\_INVALID",  "title": "The value of the attribute, that is requested to be added, is invalid."  }  ] |

when the value "def" for "attrB" is invalid, for example when "attrB" is of type "number".

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_NOT\_WRITABLE",  "title": "The attribute, that is requested to be added, is not writable."  }  ] |

when "attrB" is not writable by MnS Consumers.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_INVARIANT",  "title": "The attribute, that is requested to be added, is invariant."  }  ] |

when "attrB" is invariant and cannot be added after object creation time anymore.

"reason": ["ATTRIBUTE\_NOT\_FOUND"],

not applicable

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "REQUEST\_OBJECTS\_MISMATCH",  "reason": "NEW\_ATTRIBUTE\_PARENT\_NOT\_FOUND",  "title": "The patch operation remove is not supported."  }  ] |

when "attrA" does not exist

"reason": ["FINAL\_MV\_ATTRIBUTE\_VALUE\_INVALID"],

not applicable.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "OP\_UNKNOWN",  "title": "The patch operation remove is not supported."  }  ] |

when "add" is not supported.

**JSON Patch example 2 ("remove"):**

In the next example an attribute field is requested to be removed.

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

Error responses may look like:

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_NOT\_WRITABLE"  "title": "The attribute, that is requested to be removed, is not writable."  }  ] |

when "attrB" is not writable by MnS Consumers.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_INVARIANT",  "title": "The attribute, that is requested to be removed, is invariant."  }  ] |

when "attrB" is invariant and cannot be removed after object creation time.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "IE\_NOT\_FOUND",  "reason": "ATTRIBUTE\_NOT\_FOUND",  "title": "The attribute, that is requested to be removed, does not exist."  }  ] |

when "attrA/attrB" does not exist.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "OP\_UNKNOWN",  "title": "The patch operation remove is not supported."  }  ] |

when "remove" is not supported.

**JSON Patch example 3 ("replace"):**

In the next example the current value of an attribute field is requested to be replaced by a new value.

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

Error responses may look like:

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "OP\_UNKNOWN",  "title": "The patch operation replace is not supported."  }  ] |

when "replace" is not supported.

"reason": "NEW\_ATTRIBUTE\_NAME\_INVALID"

not applicable.

"reason": "NEW\_ATTRIBUTE\_PARENT\_NOT\_FOUND "

not applicable.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "ATTRIBUTE\_VALUE\_INVALID",  "title": "The value, that is requested to replace an existing attribute value, is invalid."  }  ] |

when the value "def" for "attrB" is invalid, for example when "attrB" is of type "number".

"reason": ["FINAL\_ATTRIBUTE\_VALUE\_INVALID"],

not applicable.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_NOT\_WRITABLE",  "title": "The attribute, whose value is requested to be replaced, is not writable."  }  ] |

when "attrB" is not writable by MnS Consumers.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 403,  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_INVARIANT",  "title": "The attribute, whose value is requested to be replaced, is invariant."  }  ] |

when "attrB" is invariant, and its value cannot be replaced after object creation time.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "IE\_NOT\_FOUND",  "reason": "ATTRIBUTE\_NOT\_FOUND",  "title": "The attribute, whose value is requested to be replaced, does not exist."  }  ] |

when "attrA/attrB" does not exist.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 422,  "type": "REQUEST\_OBJECTS\_MISMATCH"  "reason": "NEW\_ATTRIBUTE\_PARENT\_NOT\_FOUND",  "title": "The parent of the attribute field, that is requested to be added, does not exist."  }  ] |

when "attrA" does not exist.

|  |
| --- |
| ...  [  {  "badOp": "/0",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "OP\_UNKNOWN",  "title": "The patch operation add is not supported."  }  ] |

when the operation "add" is not supported.

**JSON Merge Patch example 1:**

Assume a resource where the attributes "attrA" and "attrB" exist. The following message requests to replace the current value of "attrB" with "def".

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

When "attrB" is invariant the MnS Producer might respond as follows.

|  |
| --- |
| HTTP/1.1 403 Forbidden  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "MODIFICATION\_NOT\_ALLOWED",  "reason": "ATTRIBUTE\_INVARIANT",  "title": "The attribute field, whose value is requested to be replaced, is invariant.",  "badAttributes": ["#/attributes/attrA/attrB"]  }  ] |

X.4.4 Error reasons for object manipulations

The following reasons are defined for errors that may occur when attempting to create objects with PUT, POST. 3GPP JSON Merge Patch and 3GPP JSON Patch, or when attempting to delete objects with DELETE, 3GPP JSON Merge Patch and 3GPP JSON Patch:

- OBJECT\_CREATION\_NOT\_ALLOWED (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): One or more objects cannot be created as requested because objects of this class cannot be created by MnS Consumers.

- OBJECT\_DELETION\_NOT\_ALLOWED (related type: MODIFICATION\_NOT\_ALLOWED, 403 Forbidden): One or more objects cannot be deleted as requested, because objects of this class cannot be deleted by MnS Consumers.

- NEW\_OBJECT\_CLASS\_NAME\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more objects cannot be created as requested, because the receive object class name is unknown to the MnS Producer.

- NEW\_OBJECT\_REPRESENTATION\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more objects cannot be created as requested, because the received object representation does not validate.

- NEW\_OBJECT\_CONTAINMENT\_INVALID (related type: VALIDATION\_ERROR, 400 Bad Request): One or more objects cannot be created under the specified parent as requested, because this containment is not allowed.

- NEW\_OBJECTS\_ID\_EXISTS (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Content): One or more objects cannot be created as requested, because the received "id" exists already under the specified parent.

- NEW\_OBJECTS\_PARENT\_NOT\_FOUND (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Content): One or more objects cannot be created as requested, because their specified parents do not exist.

- NEW\_OBJECT\_ATTRIBUTE\_VALUE\_MISSING (related type: VALIDATION\_ERROR, 400 Bad Request): One or more objects cannot be created as requested, because attribute or attribute field values, that shall be provided in the creation request, are not provided.

- OBJECTS\_CARDINALITY\_INVALID (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Content): One or more objects cannot be created or deleted as requested, because this would result in violating cardinality constraints.

- OBJECT\_NOT\_A\_LEAF (related type: REQUEST\_OBJECTS\_MISMATCH, 422 Unprocessable Content): One or more objects cannot be deleted as requested, because they are not leaf objects.

- OBJECT\_NOT\_FOUND (related type: IE\_NOT\_FOUND, 400 Bad Request): One or more objects cannot be deleted as requested, because they do not exist.

- OP\_UNKNOWN (related type: VALIDATION\_ERROR, 400 Bad Request): The patch operation specified by the "op" property is not known by the MnS Producer. This situation may occur, for example, when a patch operation is not supported or wrongly spelled.

The list of error reasons is not exhaustive. Other values may be defined and used as well. The name style of these enumeration literals shall follow clause 5.3.5.3 of TS 32.156 [?].

The error reason "NEW\_OBJECT\_REPRESENTATION\_INVALID" provides no information on why the representation of the resource requested to be created is invalid. A MnS Producer may decide to provide more details by specifying the error reasons related to attributes defined in clause X.4.3.2 instead of the general reason "NEW\_OBJECT\_REPRESENTATION\_INVALID". The attributes or attribute fields with problems are specified by the "badAttributes" property.

**PUT example:**

In this example a MnS Producer requests the creation of a resource using PUT.

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

When the resource representation provided in the request is invalid the MnS Producer may send the following error response.

|  |
| --- |
| HTTP/1.1 400 Bad Request  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "VALIDATION\_ERROR",  "reason": "NEW\_OBJECT\_REPRESENTATION\_INVALID",  "title": "The object cannot be created because its representation is invalid."  }  ] |

The MnS Producer may also choose to provide information on why the resource representation is invalid. For example, when an attribute name or attribute field name is invalid, the MnS Producer may return the following error response.

|  |
| --- |
| HTTP/1.1 400 Bad Request  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "VALIDATION\_ERROR",  "reason": "NEW\_ATTRIBUTE\_NAME\_INVALID",  "title": "The object representation is invalid because an attribute field name is invalid.",  "badAttributes": ["#/attributes/attrA/attrB"]  }  ] |

It is possible that the request fails for multiple reasons. For example, the object representation might be invalid, and the "id" of the resource requested to be created does already exist.

|  |
| --- |
| HTTP/1.1 207 Multi-Status  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "NEW\_OBJECT\_REPRESENTATION\_INVALID",  "title": "The object cannot be created because its representation is invalid."  },  {  "status": 422,  "type": "REQUEST\_OBJECTS\_MISMATCH",  "reason": "NEW\_OBJECTS\_ID\_EXISTS",  "title": "The object cannot be created because the object id exists already."  }  ] |

**3GPP JSON Patch example:**

Assume the following patch is applied to an object tree, that has one "SubNetwork" instance only. The first operation requests to create a "ManagedElement". This operation is successful. The second operation requests to create a "HuhuFunction" object under the new object. The "HuhuFunction" is not known to the MnS Producer. This operation fails. The third operation fails as well, since it requests to create a new object under an object that does not exist.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1  Host: example.org  Content-Type: application/3gpp-json-patch+json  [  {  "op": "add",  "path": "/ManagedElement=ME1",  "value": {  "id": "ME3",  "objectClass": "ManagedElement",  "attributes": {  "userLabel": " Berlin NW 3",  "vendorName": "Company XY",  "location": "Spandau"  }  }  },  {  "op": "add",  "path": "/ManagedElement=ME1/HuhuFunction=HUHUF1",  "value": {  "id": "XYZF1",  "objectClass": "XyzFunction",  "attributes": {  "attrA": "xyz",  "attrB": 771  }  }  },  {  "op": "add",  "path": "/ManagedElement=ME3/XyzFunction=XYZF1",  "value": {  "id": "XYZF2",  "objectClass": "XyzFunction",  "attributes": {  "attrA": "abc",  "attrB": 772  }  }  }  ] |

The error response may look like:

|  |
| --- |
| HTTP/1.1 207 Multi-Status  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "badOp": "/1",  "status": 400,  "type": "VALIDATION\_ERROR",  "reason": "NEW\_OBJECT\_CLASS\_NAME\_INVALID",  "title": "The class of the new object to be created is invalid."  },  {  "badOp": "/2",  "status": 422,  "type": "REQUEST\_OBJECTS\_MISMATCH",  "reason": "NEW\_OBJECTS\_PARENT\_NOT\_FOUND",  "title": "The parent object of the new object to be created does not exist."  }  ] |

**3GPP JSON Merge Patch example:**

Assume the "ManagedElement" with the identifier "ME3" does not exist. Then the following message requests to create two new objects under a non-existent object. This request cannot be satisfied.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1  Host: example.org  Content-Type: application/3gpp-merge-patch+json  {  "id": "SN1",  "ManagedElement": [  {  "id": "ME3",  "XyzFunction": [  {  "id": "XYZF1",  "objectClass": "XyzFunction",  "attributes": {  "attrA": "xyz",  "attrB": 771  }  },  {  "id": "XYZF2",  "objectClass": "XyzFunction",  "attributes": {  "attrA": "abc",  "attrB": 772  }  }  ]  }  ]  } |

The error message may look like:

|  |
| --- |
| HTTP/1.1 422 Unprocessable Content  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "REQUEST\_OBJECTS\_MISMATCH",  "reason": "NEW\_OBJECT\_PARENT\_NOT\_FOUND",  "title": "The object, below which new objects are requested to be created, does not exist.",  "badObjects": ["/ManagedElement=ME3"]  }  ] |

X.4.5 Error reasons for application layer errors

Error reasons for the error type "APPLICATION\_LAYER\_ERROR" are very dependent on the specific application. Therefore, it is almost impossible to define application layer error reasons that are applicable to more than one application.

This specification defines the following values for the "reason" property:

- RESOURCE\_LOCKED (related type: RETRIEVAL\_NOT\_ALLOWED ,403 Forbidden): The resource was locked by administrative action and cannot be accessed.

- SERVICE\_LOCKED (HTTP error code: 503 Service Unavailable): The MnS Producer has been locked by administrative action and is currently unable to handle the request. This condition may occur, for example, due to scheduled maintenance. The "reason" property shall be absent.

*Editor's note: When other values for "reason" are used they should follow a certain pattern (capital letters only, short, …). This aspect should be clarified here or by referencing to some general rules for string enums somewhere else.*

In the following example a MnS Consumer requests the creation of a "PerfMetricJob" instance indicating that "metric1" and "metric2" shall be collected for "obj1" and "obj2" with a granularity period if 5min.

|  |
| --- |
| PATCH /SubNetwork=SN1 HTTP/1.1  Host: example.org  Content-Type: application/3gpp-json-patch+json  [  {  "op": "add",  "path": "/PerfMetricJob=PMJ1",  "value": {  "PerfMetricJob": [  {  "id": "PMJ1",  "objectClass": "PerfMetricJob",  "objectInstance": "SubNetwork=SN1,PerfMetricJob=PMJ1",  "attributes": {  "granularityPeriod": "5",  "perfMetrics": [  "metric1",  "metric2"  ],  "objectInstances": [  "obj1",  "obj2"  ]  }  }  ]  }  }  ] |

When the requested granularity period is not supported, the "PerfMetricJob" instance is not created. The MnS Producer might answer with the following error response.

|  |
| --- |
| HTTP/1.1 400 Bad Request  Date: Tue, 06 Aug 2019 16:50:26 GMT  Content-Type: application/vnd.3gpp.error+json  [  {  "type": "APPLICATION\_LAYER\_ERROR",  "reason": "GRANULARITY\_PERIOD\_NOT\_SUPPORTED",  "title": "The requested granularity period for metric collection is not supported."  }  ] |

X.4.6 Security considerations

When the MnS Consumer is not trustworthy or the MnS Producer does not want to disclose error details, just the "type" property may be included in the error response. The response body may be omitted also completely, and just the error status code be returned in the response status line.

\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 4.6 Issue#6: Operation for multiple MOI updates (stage 2)

### 4.6.1 Issue description

The stage 2 definitions of the Provisioning MnS contain only specific operations for creation of new objects, and the deletion or manipulation of (existing) objects:

- The operation "createMOI" for requesting the creation of one object.

- The operation "deleteMOI" for requesting the deletion of one object, or of multiple objects identified by scoping and filtering.

- The operation "modifyMOIAttributes" for requesting the replacement of attribute values of one object, or of multiple objects identified by scoping and filtering.

Stage 2 does not include an operation allowing to create, delete and manipulate multiple objects using a single request. Note that stage 3 includes capabilities to create, delete and manipulate multiple objects using a single request for both HTTP/JSON and NETCONF/YANG solutions already.

### 4.6.2 Analysis

The definitions at stage 2 do not reflect the situation at stage 3. This is misleading to readers not concerned with stage 3. They may be left with the impression that the Provisioning MnS offers only limited capabilities. At stage 2 there should be an operation for creating, deleting, and manipulating multiple objects using a single request, too.

Note that no stage 3 changes are targeted by the issue in this clause.

### 4.6.3 CR proposal

As a solution for the issue, the following clause could be added to TS 28.532.

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11.1.1.4a changeMOIs operation

11.1.1.4a.1 Definition

This operation is invoked by MnS consumers to request a MnS producer to create, delete, and update multiple objects using a single request. The request contains an ordered set of sub-operations. Each sub-operation creates an object, deletes an object, or updates attribute or attribute field values. Sub-operations should be executed in the order they appear in the request.

The "baseObjectInstance" parameter is common for all sub-operations and identifies the root of the object tree where changes can be made. Each sub-operation is defined by the "path", "modifyOperator" and "nodeValue" parameters. The "path" parameter specifies the offset from the root object to the target object, the target attribute or the target attribute field of the sub-operation. The "modifyOperator" specifies the operation to be applied. Valid values are "replace", "add", remove, and for attributes and attributes fields also the value "setToDefault". The "replace" operator is not applicable when "path" identifies an object.

The "nodeValue" provides the value for the sub-operation. The parameter shall be absent for "remove" operations.

For operations on attribute values or attribute field values the same provisions as in clause 11.1.1.3 apply.

When adding (creating) objects, the "nodeValue" contains the object representation.

Note that the parameters introduced and used in this clause just serve the purpose of explaining the functionality. Specific stage 3 solutions may implement the functionality in very different ways.

11.1.1.4a.2 Input parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | S | Matching Information / Legal Values | Comment |
| baseObjectInstance | M | ManagedEntity.objectInstance | Identifies the base object, that together with the "pat" identify the |
| modificationsIn | M | LIST OF SEQUENCE <  path,  modifyOperator,  nodeValue  > | Set of sub-operations to be applied to the target node.  The "nodeIdentifier" specifies the target node.  The "modifyOperator" specifies the operation to be applied to the target attribute node. The parameter can have the values "replace", "add", "remove" or "setToDefault". The value "replace" is not applicable, when the target node is an object. The value "SetToDefault" is applicable only to attributes and attribute fields.  The "nodeValue" specifies the value for the sub-operation. This parameter is absent for "remove" operations. |

11.1.1.4a.3 Output parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | S | Matching Information / Legal Values | Comment |
| modificationsOut | M | LIST OF SEQUENCE <  objectInstance DN,  objectClass string,  LIST OF SEQUENCE<  attribute name,  attribute value >  > | Provides for each object, that is modified, the object name, the object class, and a list of name/value pairs with the values of *all* attributes after modification.  If all requested modifications are applied, the parameter may be absent.  If no requested modification is applied and an error response is returned, the parameter may be absent, too. |
| status | M | ENUM (  SUCCEEDED,  PARTIALLY\_FAILED,  FAILED  ) | Indicates if all, some or none of the requested modifications were applied. Details on the error, such as which modification could not be applied and the corresponding reason, may be returned as well. |

\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 4.6.4 Conclusion

It is recommended to add the operation "changeMOIs" as defined in clause 4.6.3 of the present document to clause 11.1.1 of TS 28.532. This requires as well the update of some mapping tables in clause 12.1 of TS 28.532, but no change of the real stage 3 functionality.

## 4.7 Issue#7: Advertising communication options (REST SS)

### 4.7.1 Issue description

TS 32.158 describes no mechanism allowing to advertise the communication options a resource supports. Communication options include supported HTTP methods, supported patch formats, and supported query parameters.

This issue applies only the HTTP/JSON solution.

### 4.7.2 Analysis

The OPTIONS method defined in IETF RFC 7231 [x], and the "Accept-Patch" header defined in IETF RFC 5789 [x] can be used to advertise the supported HTTP methods and the supported patch formats.

For advertising supported query parameters there is no method standardized yet, nor in 3GPP or other standard bodies. It is proposed to introduce the new header "Accept-Get" for that purpose.

### 4.7.3 CR proposal

The following clause shall be added to TS 32.158 [x].

\*\*\*\*\*\* BEGIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6.6 Advertising communication options

6.6.1 The HTTP OPTIONS method

The OPTIONS method is used to request communication options that are supported for the target resource identified in the OPTIONS request. A MnS Consumer can specify the target resource with a URI to refer to a specific resource on a MnS Producer, or with an asterisk ("\*") to refer to a MnS Producer in general.

The target resource must exist for an OPTIONS request to be successful, otherwise a "404 Not Found" error response shall be returned. The target URI in the OPTIONS request shall identify the URI subsequently used for sending requests. Note that this is normally not the canonical URI, as described in clause 4.4.4.

The OPTIONS method shall be supported for all resources.

*Editor's note: What is returned when the target resource is "\*" is ffs.*

6.6.2 Advertising supported HTTP request methods

The "Allow" response header (IETF RFC 7231 [2], clause 7.4.1) allows a MnS Producer to specify a comma-separated list of HTTP request methods supported for a resource.

The MnS Producer shall include an "Allow" response header (IETF RFC 7231 [2], clause 7.4.1) in the response to an OPTIONS request.

Note that, according to IETF RFC 7231 [2], clause 7.4.1, the "Allow" header shall be included as well in a "405 Method Not Allowed" response.

For example, to find out which request methods are supported by a specific "SubNetwork" instance a MnS Consumer may send the following request.

|  |
| --- |
| OPTIONS ProvMnS/v1600/SubNetwork=SN1 HTTP/1.1  Host: example.org |

The "Allow" header in the answer indicates the supported request methods.

|  |
| --- |
| HTTP/1.1 204 No Content  Allow: GET,PUT,DELETE,HEAD,OPTIONS |

6.6.3 Advertising supported patch request formats

The "Accept-Patch" response header (IETF RFC 5789 [11], clause 3.1) allows a MnS Producer to specify patch document formats accepted for a resource supporting the PATCH request method. The accepted patch formats are identified using their media types, followed by one or more optional parameters. Media types are separated by a comma.

Resources that support PATCH shall include the "Accept-Patch" header in the OPTIONS response.

A server receiving a PATCH request with an unsupported media type in the message body shall reply with a "415 Unsupported Media Type" response and include an "Accept-Patch" header referencing one or more supported patch document formats. Note that when the PATCH method is not supported, a "405 Method Not Allowed" error response shall be returned.

The patch formats that may be supported according to this specification are listed in Table 6.6.3-1.

Table 6.6.3-1: Media types for patch document formats

|  |  |
| --- | --- |
| **Media type** | **Definition** |
| application/merge-patch+json | clause 6.3.2 |
| application/json-patch+json | clause 6.3.3 |
| application/vnd.3gpp.merge-patch+json | clause 6.4.2 |
| application/vnd.3gpp.json-patch+json | clause 6.4.3 |

The following example demonstrates the use of the "Accept-Patch" header in an OPTIONS response. Assume that the target resource supports, besides other methods, the PATCH method.

|  |
| --- |
| OPTIONS ProvMnS/v1600/SubNetwork=SN1 HTTP/1.1  Host: example.org |

The "Allow" header in the answer indicates the supported request methods. The "Accept-Patch" header lists the patch formats supported for the target resource.

|  |
| --- |
| HTTP/1.1 204 No Content  Allow: GET,PUT,PATCH, DELETE,HEAD,OPTIONS  Accept-Patch: application/json-patch+json, application/vnd.3gpp.json-patch+json |

In the next example the MnS Consumer sends a patch format, that is not accepted by the MnS Producer.

|  |
| --- |
| PATCH ProvMnS/v1600/SubNetwork=SN1 HTTP/1.1  Host: example.org  Content-Type: application/merge-patch+json  [patch document] |

The error response indicates that the media type in the request is not supported and suggests to use another one.

|  |
| --- |
| HTTP/1.1 415 Unsupported Media Type  Accept-Patch: application/json-patch+json |

6.6.4 Advertising supported query parameters

This specification introduces the new "Accept-Get"response header. It is used by MnS Producers to specify the query parameters supported by the GET method on a specific resource. It is a comma-separated list of query parameter names.

Resources that support GET shall include the "Accept-Get" header in the OPTIONS response.

A server receiving a GET request with unsupported query parameters in the query component of the URI shall reply with a "400 Bad Request" response and include an "Accept-Get" header referencing all supported query parameters. Note that when the GET method is not supported, a "405 Method Not Allowed" error response shall be returned.

This specification defines the query parameters listed in Table 6.6.4-1.

Table 6.6.4-1: Query parameter names

|  |  |
| --- | --- |
| **Query parmeter name** | **Definition** |
| scopeType | clause 6.1.2 |
| scopeLevel | clause 6.1.2 |
| filter | clause 6.1.3 |
| attributes | clause 6.2.2. |
| fields | clause 6.2.2. |

The following example demonstrates the use of the "Accept-Get" header in an OPTIONS response.

|  |
| --- |
| OPTIONS ProvMnS/v1600/SubNetwork=SN1 HTTP/1.1  Host: example.org |

The response includes an "Accept-Get" header with the supported query parameter names.

|  |
| --- |
| HTTP/1.1 204 No Content  Allow: GET,PUT,PATCH, DELETE,HEAD,OPTIONS  Accept-Patch: application/json-patch+json, application/vnd.3gpp.json-patch+json  Accept-Get: attributes, fields, scopeType, scopeLevel |

\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 4.7.4 Conclusion

It is recommended to add to TS 32.158 the methods for advertising supported communication options, that are described in the CR proposal in clause 4.7.3 of the present document.

## 4.8 Issue #8: Logging

### 4.8.1 Issue description

Logging is a crucial functionality of the OAM systems. It assists in several functionalities e.g troubleshooting. The functionality of logging is available in IRP framework in TS 32.332. However, the SBMA lacks the ability of Logging.

The SBMA (service based management architecture) principle call for a management system where the MnS (management services) are defined for all management tasks e.g provisioning, performance assurance. The authorized consumer can access the MnSes as per their requirements. The current SBMA framework do not provide any mechanism to control and produce the logs for consumed management services.

The logging in IRP framework is restricted to only to logging of notifications. However, logging of Management Service require logging of all Component A, B and C i.e what operation and notifications were accessed, what were the related MIBs and what PM/KPI data was collected.

The MnS can be accessed from within the management system and from outside the management system. The logging may support both the scenarios.

### 4.8.2 Analysis

The logging mechanism for SBMA framework need to be defined.

### 4.8.3 Potential Requirements

* The 3GPP Management System shall have the capability to enable logging including recording of the operations invoked.
* The 3GPP Management System shall have the capability to enable logging including recording of the notifications produced.
* The 3GPP Management System shall have the capability to enable logging including recording of the MIBs accessed.
* The 3GPP Management System shall have the capability to enable logging including recording of the performance measurements reporting. This implies logging the measurement job contents, but excludes logging the individual measurement results.
* The 3GPP Management System shall have the capability to enable logging including recording of the KPIs produced. This implies logging the KPI generated, but excludes logging the individual KPI results.
* The 3GPP Management System shall provide a way for the consumer to retrieve the logs.
* The 3GPP Management System shall have the capability to manage log overflow situation.
* The 3GPP Management System shall include a timestamp for each log record.
* The 3GPP Management System shall include the user/consumer identity for each log record.

### 4.8.4 Potential Solutions

The potential solutions are left for the normative phase of the work.

### 4.8.5 CR proposal

The concrete CR is left for the normative phase of the work.

## 4.9 Issue#9: Harmonized approach for node selection

### 4.9.1 Issue Description

Nodes (objects, attributes, attribute fields, attribute elements) need to be selected by a MnS Consumer in many situations.

The following examples show where nodes are selected with attributes of managed objects.

- In subscriptions to alarm notifications a MnS Consumer needs to specify the objects from which he wants to get alarm notifications (using the "scope" attribute of "NtfSubscriptionControl").

- In subscriptions to CM notifications a MnS Consumer needs to specify the nodes from which he wants to get CM notifications (using the "scope" attribute of "NtfSubscriptionControl").

- When requesting measurement collection, a MnS consumer needs to specify the measured objects (using the "objectInstances" and "rootObjectInstances" attributes of "PerfMetricJob").

- When requesting threshold monitoring, a MnS consumer needs to specify the monitored objects (using the "objectInstances" and "rootObjectInstances" attributes of "ThresholdMonitor").

The following examples show where nodes are selected in operations:

- In read requests a MnS Consumer needs to specify the nodes to be returned.

Note that with NETCONF/YANG nodes are specified in the get-config operation using subtree filtering or using an XPath expression that yields a node set as output. In the HTTP/JSON solution a query parameter is defined for GET, that uses the old ITU-T defined scope mechanism for node selection, that is less powerful than XPath expressions for node selection. The node set selected by scoping can be further refined by the "filter" query parameter. This parameter uses an XPath expression.

### 4.9.2 Analysis

#### 4.9.2.1 Node selection in read operations

NETCONF/YANG supports already XPath (besides subtree filtering) for selecting nodes to be returned in get-config. No action is required.

HTTP/JSON uses scoping and filtering for node selection. Filtering cannot be used without scoping. It would be possible to select with scoping a complete subtree and filter an that subtree, but the scope could be also omitted in that case. The subtree is already defined by the target URI. For that reason, it should be possible to use the "filter" query parameter also without the "scope" query parameter.

At stage 2, the definition of the operation "getMOIAttributes" may be updated as well to include the possibility to achieve node selection based on some node selection expression only.

#### 4.9.2.2 Node selection with NRM attributes

As described in the previous clause, node selection with NRM attributes is achieved based on different mechanisms.

At stage 2 node selection could be aligned by introducing a common node selection mechanism based on some node selection notation. This node selection mechanism could be described at stage 2 only in a generic way, like the "filter" parameter in clause 11.1 of TS 32.532. The notation is stage 3 specific:

- For NETCONF/YANG XPath can be used.

- For HTTP/JSON XPath can be used.

XPath profiles may be defined.

### 4.9.3 Potential requirements

[Req-1] The read operation shall support node selection based on some node selection notation.

[Req-2] The 3GPP management system shall support one harmonized method for node selection with NRM attributes, that is based on a node selection notation.

### 4.9.4 Potential solution

**Enhancements at stage 2 for read operations:**

Clarify that a node selection mechanism based on some node selection notation is optionally supported.

**Enhancements at stage 3 for read operations:**

NETCONF/YANG: Clarify that the NETCONF XPath capability is optionally supported.

HTTP/JSON: Clarify that the "filter" query parameter can be used optionally without the "scope" query parameter for node selection.

**Enhancements at stage 2 for NRM attributes selecting nodes:**

Clarify that node selection may optionally use some node selection notation.

**Enhancements at stage 3 for NRM attributes selecting nodes:**

Clarify that an optional stage 3 solution for the node selection notation is Xpath for NETCONF/YANG and HTTP/JSON.

### 4.9.5 Conclusion

It is recommended to implement the enhancements described in clause 4.X.4 in TS 32.532, 28.622/3, and potentially also in other specifications where functionality to select nodes is specified.

## 4.10 Issue#10: OAS definition versioning (HTTP/JSON only)

### 4.10.1 Issue Description

The version of an OAS definition is currently identical to the version of the TS where the OAS definition is documented. For example, the OAS definition "TS28532\_ProvMnS.yaml" in TS 32.532 V17.5.0 has the version string "v1750".

OAS is the abbreviation for OpenAPI Specification.

Note that a MnS is composed of MnS component type A, and MnS component type B or MnS component type C. Each component may have one or more OAS definition modules.

For example, a MnS might be composed of the following modules

- TS28532\_ProvMnS.yaml

- TS28623\_GenericNRM.yaml

- TS28623\_ComDefs.yaml

- TS28.541\_NrNRM.yaml

All these modules have independent and potentially different OAS definition versions. The issue in this clause is about versioning of all OAS definition modules.

Note also that the OAS definition version is currently used in different places:

- Each OAS definition has an "info" property in its header, where the version of the module is documented.

- It is used in the target URI

- {scheme}://{URI-DN-prefix}/{root}/{MnSName}/{MnSVersion}/{URI-LDN}

- {scheme}://{authority}/{root}/{MnSName}/{MnSVersion}/{MnSResourcePath}

- The OAS definition version used in the target URI is also included as an attribute in "MnsInfo".

The version used in the target URI is for the usual MnS, based on CRUD operations and a NRM, the version of the module "TS28532\_ProvMnS.yaml", that contains the OAS definition of the CRUD operations and imports the OAS definition of the NRM.

For legacy MnS like the Fault Supervision MnS it is the version of the module "TS28532\_FaultMnS.yaml".

### 4.10.2 Analysis

The version of the OAS definition is incremented not only when the OAS definition is updated, but also when other changes to the TS are applied that have no impact on the OAS definition. This leads to a situation where the same OAS definition may have multiple version numbers. This in turn would require the introduction of a register where the information about which version numbers are identifying the same OAS definition is maintained.

A more practical solution seems to be to decouple the TS version number and OAS definition version number and to allow them to evolve independently.

### 4.10.3 Potential requirements

Potential requirements to address the issue of this clause are:

[Req-1] The numbering scheme of OAS definition shall be independent from the TS numbering scheme documenting the OAS definition.

[Req-2] The numbering scheme of OAS definition shall be based on rules for incrementing the version number that take backwards compatibility and interoperability considerations into account.

### 4.10.4 Potential solutions

The OAS definition version should be determined based on rules for incrementing the version number. These rules should take backwards compatibility and interoperability considerations into account.

### 4.10.5 CR proposal

The concrete CR is left for the normative phase of the work.

### 4.10.6 Conclusion

It is recommended to start normative work for this issue and develop rules for OAS definition versioning.

## 4.11 Issue #11: Advertising NRM properties by MnS producer

### 4.11.1 Issue description

SA5 Provisioning MnS defines NRMs with IOCs with attributes along with attribute properties and Provisioning APIs for CRUD operations. Different MnS producers may support different NRMs, for example there might be one MnS producer that is dedicated for configuration management while an another MnS producer that is dedicated for performance management. Hence the NRM properties supported by each MnS producers in a multi-vendor environment could be different, and the MnS producer implementing an NRM might support all or only a subset of the attributes for the IOCs defined in SA5. A mechanism for the MnS consumer to be aware of such the NRM properties of the MnS producer is necessary.

This key issue is to study the current situation and propose solution to provide a mechanism for the MnS producer to advertise the following NRM properties to the MnS consumers.

- Advertise the list of IOCs supported by the MnS producer.

- Advertise the attributes of each IOC that is supported by the MnS Producer.

The following aspects of an MnS producer NRM properties are analysed as part of this key issue:

- A MnS producer might not support an IOC since the underlying functionality is not supported. For example, a MnS producer that does not have the capability to support performance metrics collection and reporting would not support PerfMetricJob IOC (defined in clause 4.3.31 of TS 28.622[14]).

Note: This key issue analyses advertising the list of IOCs supported by the MnS producer. This key issue does not address a MnS producer advertising the list of instantiated MOIs.

- A MnS producer might not support all attributes (i.e., the attributes with the support qualifier defined as optional or conditional mandatory or conditional optional) of an IOC.

This clause analyses the current situation and proposes a solution where the NRM properties related schema may be advertised by MnS producer.

### 4.11.2 Current situation

Discovery of Management Services use cases and procedures are defined in clause 5 of TS 28.537[13]. It has been defined that MnS producers need to register themselves with their management capabilities in the 3GPP management system. The data describing a MnS producer, and their capabilities is called MnS information or MnS profile.

The related NRMs MnSRegistry and MnsInfo is defined in clause 4.3.42 and 4.3.41 of TS 28.622[14] respectively. MnSInfo hold the information related to the MnS label (attribute mnsLabel), type (attribute mnsType), version (attribute mnsVersion), address (attribute mnsAddress) and scope (attribute mnsScope).

The MnS producer profile is defined in clause 4.2.4 of TS 28.533[15]. This is described as follows:

-- Start of quote from TS 28.533[15] clause 4.2.4:

“

*4.2.4 MnS producer profile*

A MnS producer is described by a set of meta data called MnS producer profile. The profile holds information about the supported MnS components and their version numbers. This may include also information about support of optional features. For example, a read operation on a complete subtree of managed object instances may support applying filters on the scoped set of objects as optional feature. In this case the MnS profile should include the information if filtering is supported.

”

-- End of quote from TS 28.533[15] clause 4.2.4 --

These existing mechanisms however do not define the mechanism for the MnS consumer to be aware of the MnS producer NRM properties described in clause 4.x.1, and hence a mechanism to advertise this needs to be studied and specified.

Further, any MnS producer using YANG-Netconf solution set are required to use the YANG Library (ietf-yang-library defined in reference [16]) to advertise supported IOCs, attributes and constrains to the MnS consumers.

However, any MnS producer based on OpenAPI solution set does not advertise the supported IOCs, attributes and constrains to the MnS consumers. The MnS consumers may expect that the complete NRM defined in SA5 is supported by every MnS producer. There is no mechanism currently defined where the MnS consumer can get this information. Hence, a mechanism to advertise such NRM properties in OpenAPI needs to be studied and specified.

### 4.11.3 Analysis

### 4.11.4 Potential requirements

Potential requirements to address the issue are:

[Req-1] MnS producer shall be able to advertise supported IOCs to the MnS consumers.

[Req-2] MnS producer shall be able to advertise supported attributes for the supported IOCs to the MnS consumers.

### 4.11.5 Potential solution

4.11.5.1 Introduction

Solution for potential requirements in clause 4.11.4 needs the MnS producer to advertise the following NRM properties to the MnS consumers:

- list of IOCs supported by the MnS producer.

- list of supported attributes of each IOC that is supported by the MnS Producer.

The MnS producer may advertise the address (URL) from where the MnS consumer can retrieve the advertised NRM properties for each MnS that the producer supports.

4.11.5.2 Extension to MnSInfo with address information to retreive NRM properties

MnSInfo IOC (defined in TS 28.622 clause 4.3.42) represents an available Management Service (MnS) and provides the data required to support its discovery. This IOC has information is used by the consumer to discover the producers of specific Management Services and to derive the addresses of the Management Service.

This potential solution proposes to enhance MnSInfo with a new mnsAddressForNrmProperties attribute. Attribute mnsAddressForNrmProperties is used to provide addressing information for the NRM properties schema of the Management Service producer.

Following table describe the description of mnsAddressForNrmProperties attribute proposed for MnSInfo IOC.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute name | S | isReadable | isWritable | isInvariant | isNotifyable |
| mnsAddressForNrmProperties | M | T | F | F | T |

| Attribute Name | | Documentation and Allowed Values | | Properties | |
| --- | --- | --- | --- | --- | --- |
| mnsAddressForNrmProperties | | Addressing information for the schema containing NRM properties of the Management Service producer. | | type: String  multiplicity: 0..1  isOrdered: N/A  isUnique: N/A  defaultValue: None  isNullable: False | |

4.11.5.3 MnS Producer based on OpenAPI solution set

This clause proposes a mechanism for MnS producer, based on OpenAPI solution set, to advertise the supported NRM properties to the MnS consumer.

The MnS producer advertises the schema, with the OpenAPI description in the YAML file format, that it supports, along with the information related to the corresponding 3GPP defined standard schema. The schema advertised by the MnS producer contains the IOCs and attributes supported. The MnS consumer(s) requiring this information can query this schema from the URL (in MnSInfo attribute mnsAddressForNrmProperties) to retrieve the information about the NRM properties supported by the MnS producer.

This potential solution does not propose the schema advertised by the MnS producer to advertise the information related to supported cardinality, name-containment, and vendor extensions.

4.11.5.4 MnS Producer based on YANG-Netconf solution set

Clause 6.2.1.5 of TS 32.160 [17] specifies that the MnS producer using YANG-Netconf solution set uses YANG version 1.1. YANG 1.1 (RFC7950 [18]) requires support for YANG Library (ietf-yang-library defined in RFC8525 [16]). Hence the MnS producer using YANG-Netconf solution set is required to advertise supported IOCs, attributes and constrains or deviations, from the 3GPP defined standard NRM schema, to the MnS consumers using the YANG Library (ietf-yang-library defined in RFC8525 [16]).

### 4.11.6 CR proposal

The concrete CR is left for the normative phase of the work.

### 4.11.7 Conclusion

It is recommended to implement the enhancements described in clause 4.11.5 in TS 28.622, 28.623, and potentially also in other specifications where templates related to YANG SS and OpenAPI SS definitions are defined.

## 4.12 Issue #12: Multiple alarms subscription mechanism

### 4.12.1 Issue description

As per the existing mechanism, a consumer can subscribe to receive a newly raised alarm as part of "notifyNewAlarm" notification using two alternate mechanism a) using subscribe operation defined in clause 11.2.1.1.1 of TS 28.532 b) using "NtfSubscriptionControl" defined in clause 4.3.22 of TS 28.622.

It is confusing for the industry to have alternate subscription mechanism to get the same notification. It also entails burden on the implementers when they are asked to support both the mechanism.

### 4.12.2 Analysis

The subscribe/notify mechanism defined in 28.532 is not needed.

### 4.12.3 Potential Requirements

The 3GPP Management System shall have a single mechanism to subscribe for the alarms and get notifications accordingly.

### 4.12.2 Potential Solutions

Deprecate or remove (make it Void.) clause 11.2.1.1.1 of TS 28.532 defining operation subscribe. (Make it Void.)

Deprecate or remove (make it Void.) clause 11.2.1.1.2 of TS 28.532 defining operation unsubscribe. (Make it Void.)

### 4.12.3 Change Request proposal

The concrete change request is left for the normative phase of the work.

## 4.13 Issue #13: Partial success for update operations (HTTP/JSON only)

### 4.13.1 Issue description

TS 32.158 describes numerous HTTP methods for manipulating attributes and objects (CRUD operations). All these methods share a common property: they are atomic. This means that the update request is either satisfied in its entirety or not at all. Partial success is not supported.

This issue refers only to the HTTP/JSON solution. For NETCONF the error-option parameter allows to request partial success modes, though most implementations support only the rollack-on-error behavior.

### 4.13.2 Analysis

Partial success of update operations is currently not supported for HTTP/JSON, though stage 2 definitions in TS 28.532 require that. Partial success means that some updates in the request are satisfied, and some are not satisfied.

Processing on the MnS producer may follow three modes:

- atomic: When no error occurs, all modifications are applied. When an error occurs, no modification is applied. In case some modifications were already applied when an error is encountered, the configuration needs to be rolled back to the state prevailing before processing of the request started.

- stop-on-error: Processing of the request stops when the first error is encountered. The first modification resulting in an error and all remaining modifications are not applied. All other modifications are applied

- best effort: The complete request is processed until the end. Modifications, that can be applied, are applied. Modifications, that cannot be applied, are not applied.

For the stop-on-error no use cases are seen.

### 4.13.3 Potential requirements

Potential requirements to address the issue of this clause are:

[Reg-1] The MnS producer shall support atomic mode and should support best-effort mode of processing.

[Reg-2] A MnS consumer shall be able to indicate to a MnS producer if the update request shall be processed in atomic or best effort mode. If the requested mode is not supported, the MnS producer shall return an error response. If the same operation is used for atomic and best effort mode and no mode is indicated in the request, the default shall be atomic handling.

[Reg-3] The response to update requests processed in best effort mode shall include detailed information on which parts of the update request failed and which parts succeeded.

[Req-4] The configuration after processing the update request shall be valid according to the NRM model and related constraints in both atomic and best effort mode.

### 4.13.4 Potential solutions

HTTP Patch is defined by IETF as atomic method. It is recommended not to change this basic property. The same is true for HTTP Put. Operations for attribute and object manipulations supporting a best effort mode could be implemented with HTTP Post. The request body specifying the requested attribute and object manipulations could be inspired by or fully aligned to JSON Patch.

The error response format specified in clause 4.5 for atomic operations allows to report problem details for each sub-operation that failed. It is at the MnS producer discretion to report all failed sub-operations or only one failed sub-operation. For the best effort mode, the MnS producer needs to include all failed sub-operations in the error response. If a suboperation is not included, that indicates it was successful.

### 4.13.5 CR proposal

The concrete CR is left for the normative phase of the work.

### 4.13.6 Conclusion

It is recommended to start normative work for this issue and to add support for partial success to the UPDATE operations of the HTTP/JSON solution.

# 5 Recommendations and conclusions

Conclusions are given for each issue above.

Annex A:  
Profiles

# A.1 EBNF for basic XPath profile

LocationPath ::= AbsoluteLocationPath

AbsoluteLocationPath ::= Step

| AbsoluteLocationPath '/' Step

Step ::= AxisSpecifier NodeName Predicate\*

AxisSpecifier ::= AxisName '::'

|

AxisName ::= 'descendant'

| 'child'NodeName ::= '\*'

| Name

Predicate ::= '[' 'id' '=' Literal ']'

Literal ::= '"' [^"]\* '"'

| "'" [^']\* "'"Name ::= NameStartChar (NameChar)\*

NameChar ::= NameStartChar | "-" | "." | [0-9]

NameStartChar ::= [A-Z] | "\_" | [a-z]

# A.2 EBNF for advanced XPath profile

LocationPath ::= RelativeLocationPath

| AbsoluteLocationPath

AbsoluteLocationPath ::= '/' RelativeLocationPath?

| '//' RelativeLocationPath

RelativeLocationPath ::= Step

| RelativeLocationPath '/' Step

| RelativeLocationPath '//' Step

Step ::= AxisSpecifier NodeName Predicate\*

AxisSpecifier ::= AxisName '::'

|

AxisName ::= | 'descendant'

| 'child'

NodeName ::= '\*'

| Name

Predicate ::= '[' PredicateExpr ']'

PredicateExpr ::= EqualityExpr

| InEqualityExpr

| PathExpr

EqualityExpr ::= PathExpr '=' Literal

InEqualityExpr ::= PathExpr '<' Number

| PathExpr '>' Number

| PathExpr '<=' Number

| PathExpr '>=' Number

PathExpr ::= LocationPath

| FunctionCall

FunctionCall ::= FunctionName '(' ( Argument ( ',' Argument )\* )? ')'

Argument ::= EqualityExpr

| InEqualityExpr

| PathExpr

| Number

| Literal

Literal ::= '"' [^"]\* '"'

| "'" [^']\* "'"

Number ::= Digits ('.' Digits?)?

| '.' Digits

Digits ::= [0-9]+

FunctionExcluded ::= 'position()'

| 'last()'

| 'comment'

| 'processing-instruction'

| 'node'

FunctionName ::= Name - FunctionExcluded

Name     ::= NameStartChar (NameChar)\*

NameChar ::= NameStartChar | "-" | "." | [0-9]

NameStartChar ::= [A-Z] | "\_" | [a-z]

Annex B:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2022-04 | SA5#142e | S5-222520 | - | - | - | Initial skeleton | 0.0.0 |
| 2022-04 | SA5#142e | S5-222173 |  |  |  | Rel-18 pCR 28.831 Add skeleton | 0.1.0 |
| 2022-04 | SA5#142e | S5-222174 |  |  |  | Rel-18 pCR 28.831 Add key issue 1 Schema for notifyMOIChanges | 0.1.0 |
| 2022-04 | SA5#142e | S5-222175 |  |  |  | Rel-18 pCR 28.831 Add key issue 2 Targeted notification subscriptions | 0.1.0 |
| 2022-04 | SA5#142e | S5-222176 |  |  |  | Rel-18 pCR 28.831 Add description of current situation for key issue 2 Targeted notification subscriptions | 0.1.0 |
| 2022-05 | SA5#143 | S5-223582 |  |  |  | Rel-18 pCR 28.831 Improve description of current situation for key issue 2 (Targeted notification subscription) | 0.2.0 |
| 2022-07 | SA5#144 | S5-224424 |  |  |  | pCR 28.831 Add potential requirements for key issue Targeted notification subscription | 0.3.0 |
| 2022-07 | SA5#144 | S5-224425 |  |  |  | pCR 28.831 Add potential solutions for key issue Targeted notification subscription | 0.3.0 |
| 2022-07 | SA5#144 | S5-224426 |  |  |  | pCR 28.831 Add analysis of JSONPointer as potential solution | 0.3.0 |
| 2022-07 | SA5#144 | S5-224427 |  |  |  | pCR 28.831 Add analysis of JSONPath as potential solution | 0.3.0 |
| 2022-08 | SA5#145 | S5-225806 |  |  |  | pCR 28.831 Modify potential requirements for key issue Targeted notification subscription | 0.4.0 |
| 2022-08 | SA5#145 | S5-225805 |  |  |  | pCR 28.831 Add XPath as potential solution | 0.4.0 |
| 2022-08 | SA5#145 | S5-225804 |  |  |  | pCR 28.831 Add special XPath considerations for JSON and YANG | 0.4.0 |
| 2022-08 | SA5#145 | S5-225803 |  |  |  | pCR 28.831 Add simple XPath profiles | 0.4.0 |
| 2022-08 | SA5#145 | S5-225801 |  |  |  | pCR 28.831 Add key issue Definition of createMOI | 0.4.0 |
| 2022-11 | SA5#146 | S5-226556 |  |  |  | Rel-18 pCR 28.831 Add approved but not implemented Annex A on XPath profiles | 0.5.0 |
| 2022-11 | SA5#146 | S5-226950 |  |  |  | Rel-18 pCR 28.831 Improve revised definition of creatMOI | 0.5.0 |
| 2022-11 | SA5#146 | S5 226951 |  |  |  | Rel-18 pCR 28.831 Add issue – Definition of modifyMOIAttributes | 0.5.0 |
| 2022-11 | SA5#146 | S5-226974 |  |  |  | Rel-18 pCR 28.831 Add issue – Definition of HTTP error responses | 0.5.0 |
| 2023-03 | SA5#147 | S5-232353 |  |  |  | Rel-18 CR 28.831 Add issue on advertising communication options (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-232356 |  |  |  | Rel-18 CR 28.831 Add general solution for advertising communication options (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-232361 |  |  |  | Rel-18 CR 28.831 Add solution for advertising supported patch request formats (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-232362 |  |  |  | S5-232362 Rel-18 CR 28.831 Add solution for advertising supported query parameters (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-233036 |  |  |  | Rel-18 CR 28.831 Add conclusion for issue 4 Definition of modifyMOIAttributes | 0.6.0 |
| 2023-03 | SA5#147 | S5-233037 |  |  |  | Rel-18 CR 28.831 Add issue on Bulk CM operations (stage 2) | 0.6.0 |
| 2023-03 | SA5#147 | S5-233038 |  |  |  | Rel-18 CR 28.831 Add CR proposal and conclusion for issue on Bulk CM operations (stage 2) | 0.6.0 |
| 2023-03 | SA5#147 | S5-233045 |  |  |  | Rel-18 CR 28.831 Add solution for advertising supported HTTP request methods (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-233047 |  |  |  | Rel-18 CR 28.831 Add conclusion for issue on advertising communication options (REST SS) | 0.6.0 |
| 2023-03 | SA5#147 | S5-233048 |  |  |  | Rel-18 CR 28.831 Add CR proposal for error response – Supported HTTP error codes | 0.6.0 |
| 2023-03 | SA5#147 | S5-233049 |  |  |  | Rel-18 CR 28.831 Add CR proposal for error response – Error response body format | 0.6.0 |
| 2023-03 | SA5#147 | S5-233050 |  |  |  | Rel-18 CR 28.831 Add CR proposal for error response – Type property | 0.6.0 |
| 2023-03 | SA5#147 | S5-233051 |  |  |  | Rel-18 CR 28.831 Add CR proposal for error response – Reason property | 0.6.0 |
| 2023-03 | SA5#147 | S5-233052 |  |  |  | Rel-18 CR 28.831 Clarify the mapping from a JSON document into the XPath data model | 0.6.0 |
| 2023-03 | SA5#147 | S5-233053 |  |  |  | pCR 28.831 Logging Key Issues | 0.6.0 |
| 2023-04 | SA5#148 | S5-233285 |  |  |  | Rel-18 CR 28.831 Add definition of HTTP error codes (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233593 |  |  |  | Add a new key issue for Advertising NRM properties by MnS producer | 0.7.0 |
| 2023-04 | SA5#148 | S5-233594 |  |  |  | Add potential requirements for KI to Add a new key issue for Advertising NRM properties by MnS producer | 0.7.0 |
| 2023-04 | SA5#148 | S5-233595 |  |  |  | Rel-18 CR 28.831 Improve description of error response body format (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233596 |  |  |  | Rel-18 CR 28.831 Clarify definition of the type property (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233597 |  |  |  | 233288 Rel-18 CR 28.831 Clarify definition of the reason property for GET (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233598 |  |  |  | Rel-18 CR 28.831 Clarify definition of the reason property for manipulating attributes (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233599 |  |  |  | Rel-18 CR 28.831 Clarify definition of the reason property for manipulating objects (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233600 |  |  |  | Rel-18 CR 28.831 Add clause on Error reasons for application layer errors (error response format) | 0.7.0 |
| 2023-04 | SA5#148 | S5-233601 |  |  |  | Rel-18 CR 28.831 Add key issue on node selection | 0.7.0 |
| 2023-04 | SA5#148 | S5-233602 |  |  |  | Rel-18 CR 28.831 Add CR proposal for issue 2 Targeted notification subscriptions | 0.7.0 |
| 2023-04 | SA5#148 | S5-233603 |  |  |  | Rel-18 CR 28.831 Add key issue and solution on MnS versioning | 0.7.0 |
| 2023-05 | SA5#149 | S5-234527 |  |  |  | pCR 28.831 multiple alarms in single notifyNewAlarm Key Issue | 0.8.0 |
| 2023-05 | SA5#149 | S5-234528 |  |  |  | Rel-18 pCR 28.831 Add conclusion for issue on createMOI | 0.8.0 |
| 2023-05 | SA5#149 | S5-234529 |  |  |  | Add a potential solution for KI for Advertising NRM properties by the MnS producer | 0.8.0 |
| 2023-05 | SA5#149 | S5-234533 |  |  |  | Rel-18 pCR 28.831 Add issue on partial success for WRITE operations (HTTP/JSON only) | 0.8.0 |
| 2023-05 | SA5#149 | S5-234534 |  |  |  | Rel-18 pCR 28.831 Clarify description of the HTTP error response | 0.8.0 |
| 2023-06 | SA#100 | SP-230636 |  |  |  | Presented for information and approval | 1.0.0 |
| 2023-06 | SA#100 |  |  |  |  | Upgrade to change control version | 18.0.0 |