TM Forum Specification

TMF630 REST API Design Guidelines Part 5

JSON patch extensions to manage arrays

TMF630

Team Approved Date: Nov. 25th, 2020

Release Status: Pre-Production	**Approval Status:** Team Approved
Version: 4.0.0	**IPR Mode:** RAND

NOTICE

Direct inquiries to the TM Forum office:

181 New Road, Suite 304
Parsippany, NJ 07054, USA
Tel No. +1 973 944 5100
Fax No. +1 973 998 7916
TM Forum Web Page: www.tmforum.org

Table of Contents

Executive Summary

This document, “REST API Design Guidelines Part 5” provides information for the development of TM Forum APIs using REST.

It provides recommendations and guidelines on JSON Patch extension to manage arrays.

JSON Patch extension to manage arrays

Introduction

JavaScript Object Notation (JSON) [RFC4627] is a common format for the exchange and storage of structured data. HTTP PATCH [RFC5789] extends the Hypertext Transfer Protocol (HTTP) [RFC2616] with a method to perform partial modifications to resources.

JSON Patch [RFC6902] is a format for expressing a sequence of operations to apply to a target JSON document for use with the HTTP PATCH method.

The use of JSON Patch to make partial updates to a JSON document or to a data structure where the update affects a specific element within an array requires indicating (via array index pointer) the position that the impacted element holds within the array.

JSON Patch Query is a format that extends JSON Patch in order to handle partial updates of elements within an array without previous knowledge of the order in which the impacted elements are located within the array.

Using JSON Patch Query there is no need to refer to indexes within the array but use the contents of the different parameters defining the structure as the reference to identify each individual element within the array.

Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Document Structure

A JSON Patch Query document follows the same structure as a JSON Patch [RFC6902] document, which represents an array of objects, each object representing a single operation to be applied to the target JSON document, but the "path" member includes a query parameter to allow identifying uniquely the element within an array that is impacted by the operation.

Due to the additional functionality provided by the current document, the JSON Patch document mime type is defined as: “application**/**json-patch-query+json**”**

The following is an example JSON Patch Query document, transferred in an HTTP PATCH request:

```
PATCH /my/data HTTP/1.1
Host: example.org
Content-Length: 326
Content-Type: application/json-patch+query

[
    {"op": "remove", "path": "/a/b/c?elemInC=value" },
    { "op": "replace", "path": "/a/b/c"?elemInC=value, "value": 42 }
]
```

Evaluation of a JSON Patch Query document is performed in the same way as for a JSON Patch document.

Refer to Examples section below for a list of examples of the use of JSON Patch Query document for partial update of a resource.

With the addition of the Design Guidelines 6 - JSON Patch, the JSON Patch Query has been updated to support the addition selector called “filter” allowing the enhancement of the query with the JSON Path syntax.

The following is an example of the JSON Patch Query document, transferred in an HTTP PATCH request, where the query string takes as value the “filter” selector with the a JSONPath expression:

```
[
    {
        "op": "add",
        "path": "note[?(@.author=='John Doe')] ",
        "value": {text:Informed}
    }
]
```

For further information on JSON Path selectors see Design Guidelines 6 and further examples below in the document.

Operations

Operation in a JSON Patch Query document are handled in the same way as for a JSON Patch document but the query parameter in the “path” member is used to identify the element within an array that is impacted by the operation, the one that matches all the criteria included in the query expression.

When the query is not required because the array index can be used, this format is handled in the same way as JSON Patch [RFC6902].

Refer to Examples section below for a list of examples of the use of JSON Patch Query document for different operations on a resource.

Error Handling

Same as for a JSON Patch document, if a normative requirement is violated by a JSON Patch Query document, or if an operation is not successful, evaluation of the document SHOULD terminate and application of the entire patch document SHALL NOT be deemed successful.

See [RFC5789], Section 2.2 for considerations regarding handling errors when JSON Patch is used with the HTTP PATCH method, including suggested status codes to use to indicate various conditions.

In addition to the above, if the “filter” selector is being used in the query string and the expression fails at the syntax level a 400 Bad Request MUST be returned.If the JSON Path is not supported by the server implementation, but the client is making use of it in the “filter” selector a 501 Not Implemented MUST be returned

Examples

This section presents a set of examples of different PATCH update operations using JSON Patch document, based on real TM Forum APIs published at the time of producing this guideline.

Example 1 - Adding an attribute to one of the components of an array

Considering a Ticket resource such as `{TroubleTicket-apiRoot}/troubleTicket/{IDtt1}`

```
{
    "id": "1",
    "correlationId": "TT53482",
    ...
    "note": [{
            "date": "2013-07-24T09:55:30.0Z",
            "author": "Arthur Evans"
        },
        {
            "date": "2013-07-25T08:55:12.0Z",
            "author": "John Doe"
        }]
}
```

A JSON Patch document such as

```
[
    {
        "op": "add",
        "path": "/note/text?note.author=John Doe",
        "value":"Informed"
    }
]
```

Meaning: add to the note structure, to the component within the array whose attribute *author* has a value of “John Doe” (the second component within the array), a new attribute with name *text* and value “Informed”

would result in the following updated Ticket resource

```
{
    "id": "1",
    "correlationId": "TT53482",
    ...
    "note": [{
            "date": "2013-07-24T09:55:30.0Z",
            "author": "Arthur Evans"
        },
        {
            "date": "2013-07-25T08:55:12.0Z",
            "author": "John Doe",
            "text": "Informed "
        }]
}
```

Example 2 - Removing one of the components of an Array Element (the whole structure)

Considering a Ticket resource such as `{TroubleTicket-apiRoot}/troubleTicket/{IDtt1}`

```
{
    "id": "1",
    "correlationId": "TT53482",
    ...
    "note": [{
            "date": "2013-07-24T09:55:30.0Z",
            "author": "Arthur Evans",
            "text": "Already called the expert"
        },
        {
            "date": "2013-07-25T08:55:12.0Z",
            "author": "John Doe",
            "text": "Informed "
        },
        {
            "date": "2013-07-25T07:55:12.0Z",
            "author": "Diego Salas",
            "text": "Resolved issue"
        }]
}
```

A JSON Patch document such as

```
[
   { "op": "remove", "path": "/note?note.author=John Doe" }
]
```

Meaning: remove from the note structure the component within the array whose attribute *author* has a value of “John Doe” (the second component within the array)

would result in the following Ticket resource

```
{
    "id": "1",
    "correlationId": "TT53482",
    ...
    "note": [{
            "date": "2013-07-24T09:55:30.0Z",
            "author": "Arthur Evans",
            "text": "Already called the expert"
        },
        {
            "date": "2013-07-25T07:55:12.0Z",
            "author": "Diego Salas",
            "text": "Resolved issue"
        }]
}
```

Example 3 - Removing an attribute from one of the components of an array

Considering a Product resource such as /{ProductInventory-apiRoot}/product/{IDprd1}

```
{
    "id": "4501",
    "description": "This product ….  ",
    ...
    "productPrice": [{
            "name": "Regular Price",
            "priceType": "recurring",
            ...
            "prodPriceAlteration": {
                "name": "Shipping Discount ",
                "description": "This prod price alteration … ",
            },
            "price": { ... }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            "price": { ... }
        }]
}
```

A JSON Patch document such as

```
[
    {
        "op": "remove",
        "path": "/productPrice/prodPriceAlteration?prodPrice.name=Regular Price"
    }
]
```

Meaning: remove from the product structure, from the component within the array whose attribute *name* has a value of “Regular Price” (the first component within the array), the attribute *prodPriceAlteration*

would result in the following Product resource

```
{
    "id": "4501",
    "description": "This product …. ",
    ...
    "productPrice": [{
            "name": "Regular Price",
            "priceType": "recurring",
            ...
            price: { ... }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            price:{ ... }
        }]
}
```

Example 4 - Removing a complete complex structure component of an array

Considering a Product resource such as `/{ProductInventory-apiRoot}/product/{IDprd1}`

```
{
    "id": "4501",
    "description": "This product ….  ",
    ...
    "productPrice": [{
            "name": "Setup Price",
            "priceType": "one time",
            ...
            price: { ... }
        },
        {
            "name": "Regular Price",
            "priceType": "recurring",
            ...
            "prodPriceAlteration: {
                "name": "Shipping Discount ",
                "description": "This prod price alteration … ",
            ...
            },
            price:\{…}
        }]
}
```

A JSON Patch document such as

```
 [
 { "op": "remove", "path": "/productPrice? productPrice.name=Setup
Price" }
 ]
```

Meaning: remove from product structure the component within the array whose attribute *name* has a value of “Setup Price” (the first component within the array)

would result in the following Product resource

```
{
    "id": "4501",
    "description": "This product ….  ",
    ...
    "productPrice": [{
            "name": "Regular Price",
            "priceType": "recurring",
            ...
            "price": { ... }
        }]
}
```

Example 5 - Replacing an attribute from one of the components of an array

Considering a ProductOffering resource such as /{ProductCatalog-apiRoot}/productOffering{IDoff1}

```
{
    "id": "42",
    "description": "Virtual Storage Medium",
    "lifecycleStatus": "Active ",
    ...
    "productOfferingPrice": [{
            "name": "Monthly Price",
            "priceType": "recurring",
            ...
            "price":{
                "amount":12,
                "units":"EUR"
            }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            "price":{
                "amount":30,
                "units":"EUR"
            }
        }]
}
```

A JSON Patch document such as

```
 [
 { "op": "replace", "path": "/productOfferingPrice/price/amount?
productOfferingPrice.name=Monthly Price", "value": "25" }
 ]
```

Meaning: update the attribute named *amount* in productOffering structure for the component within the array whose attribute *name* has a value of “Monthly Price” (the first component within the array), the new value of *amount* must be set to 25

would result in the following ProductOffering resource

```
{
    "id": "42",
    "description": "Virtual Storage Medium",
    "lifecycleStatus": "Active ",
    ...
    "productOfferingPrice": [{
            "name": "Monthly Price",
            "priceType": "recurring",
            ...
            "price":{
                "amount":25,
                "units":"EUR"
            }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            "price":{
                "amount":30,
                "units":"EUR"
            }
        }]
}
```

Example 6 - Replacing a complete component of an array

Considering a ProductOffering resource such as `/{ProductCatalog-apiRoot}/productOffering/{IDoff1}`

```
{
    "id": "42",
    "description": "Virtual Storage Medium",
    "lifecycleStatus": "Active ",
    ...
    "productOfferingPrice": [{
            "name": "Monthly Price",
            "priceType": "recurring",
            ...
            "price":{
                "amount":12,
                "units":"EUR"
            }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            "price":{
                "amount":30,
                "units":"EUR"
            }
        }]
}
```

A JSON Patch document such as

```
[
    {
        "op": "replace",
        "path": "/productOfferingPrice/price?productOfferingPrice.name=Setup
Price",
        "value": {"amount":"40","units":"USD" }
    }
]
```

Meaning: update the *price* structure, the component within the array whose attribute *name* has a value of “Setup Price” (the second component within the array), replacing the complete structure of that component with the new one provided

would result in the following ProductOffering resource

```
{
    "id": "42",
    "description": "Virtual Storage Medium",
    "lifecycleStatus": "Active ",
    ...
    "productOfferingPrice": [{
            "name": "Monthly Price",
            "priceType": "recurring",
            ...
            "price":{
                "amount":12,
                "units":"EUR"
            }
        },
        {
            "name": "Setup Price",
            "priceType": "one time",
            ...
            "price":{
                "amount":40,
                "units":"USD"
            }
        }]
}
```

Example 7 - Replacing an attribute from one of the components of a complex array (resolving ambiguities)

Considering a ProductOrder resource such as `/{ProductOrdering-apiRoot}/productOrder/{IDord1}`

```
{
    "id": "3774",
    "description": "This product order covers ⋯ ",
    "requestedCompletionDate": "2017-07-14",
    ...
    "orderItem": [{
            "action": "add ",
            "quantity": 1,
            "productOffering": {
                "href":"/productOffering/1513",
                "id":"1513",
                "name":"Offer Good Plan"
            }
            ...
            "product": {
                "relatedParty":[{
                        "name":"Mary",
                        "role":"customer"
                }]
            }
        },
        {
            "action": "add ",
            "quantity": 1,
            "productOffering":{
                "href":"/productOffering/1513",
                "id":"1513",
                "name":"Offer Good Plan"
            }
            ...
            "product":{
            "relatedParty":[{
                    "name":"John",
                    "role":"customer"
                }]
            }
        }]
}
```

A JSON Patch document such as

```
[
    {
        "op": "replace",
        "path": "/orderItem/quantity?orderItem.productOffering.id=1513
&orderItem.product.relatedParty.role
=customer&orderItem.product.relatedParty.name=Mary",
        "value": "25"
    }
]
```

Meaning: update the *quantity* parameter in the *orderItem* structure, for the component within the array whose attribute *productOffering.id* has a value of “1513” but also whose attributes *product.relatedParty.role* and *product.relatedParty.name* have a value of “customer” and “Mary” (the first component within the array of orderItems)), the new value of *quantity* must be set to “25”.

In this case there are two components within orderItem array with attribute *productOffering.id* set to “1513”, therefore an additional detail is required to identify the specific attribute impacted by the operation

would result in the following ProductOrder resource

```
{
    "id": "3774",
    "description": "This product order covers ⋯ ",
    "requestedCompletionDate": "2017-07-14",
    ...
    "orderItem": [{
            "action": "add ",
            "quantity": 25,
            "productOffering":{
                "href":"/productOffering/1513",
                "id":"1513",
                "name":"Offer Good Plan"
            }
            ...
            "product": {
                "relatedParty":[{
                        "name":"Mary",
                        "role":"customer"
                }]
            }
        },
        {
            "action": "add ",
            "quantity": 1,
            "productOffering":{
                "href":"/productOffering/1513",
                "id":"1513",
                "name":"Offer Good Plan"
            }
            ...
            "product":{
                "relatedParty":[{
                        "name":"John",
                        "role":"customer"
                }]
            }
        }]
}
```

JSON Patch with “filter” selector (JSON Path)

Considering a troubleTicket resource such as /api/troubleTicket/1:

```
{
    "id": "1",
    "note": [{
            "date": "2013-07-25T06:55:12.0Z",
            "author": "John Doe",
            "status": "Edited"
        },
        {
            "date": "2013-07-24T09:55:30.0Z",
            "author": "Arthur Evans",
            "status": "Edited"
        },
        {
            "date": "2013-07-25T08:55:12.0Z",
            "author": "John Doe",
            "status": "Archived"
        }]
}
```

The existing specification enables a JSON Patch document such as following where the condition is based on a simple path with a query expression:

```
[
    {
        "op": "add",
        "path": "/note?note.author=John Doe",
        "value": "Informed"
    }
]
```

The JSON Path “filter” selector documented in Design Guidelines Part 6 will enhance the query capabilities by augmenting them with the JSON Path syntax.

Example 1 - Adding an attribute to one of the components of an array

Add where the "note" array has an "author" called John Doe

```
[
    {
        "op": "add",
        "path": "note[?(@.author=='John Doe')] ",
        "value": {"text":"Informed"}
    }
]
```

Applying the above path to the sample JSON will result in the following path to be automatically computed: "$['note'][1]"

Example: add where the "author" is “John Doe” and the “status” is “Edited”

```
[
    {
        "op": "add",
        "path": "note[?(@.author=='John Doe' && @.status=='Edited')]",
        "value": {"text":"Informed"}
    }
]
```

All the other capabilities are also supported through the JSON Path mechanisms either by a simple path, by a predicate expression or by a combination of them:

Example 2 - Removing one or multiple elements of an array where the JSON Path predicate condition is being satisfied

Example:

```
[
    {
        "op": "remove",
        "path": "note[?(@.author='John Doe')]"
    }
]
```

Note: JSON Path capabilities on arrays are described in the **"Operators"** section of the Design Guidelines Part 6.

Example 3 - Removing one element of an array based on the JSON Path expression

Example:

```
[
    {
        "op": "remove",
        "path": "note[?(@.author='John Doe')].date"
    }
]
```

Administrative Appendix

This Appendix provides additional background material about the TM Forum and this document. In general, sections may be included or omitted as desired; however, a Document History must always be included.

Document History

Version History

This section records the changes between this and the previous document version as it is edited by the team concerned. Note: this is an incremental number which does not have to match the release number and used for change control purposes only.

Version Number	Date Modified	Modified by:	Description of changes
1.0	23-Nov-2014	Pierre Gauthier TM Forum	Description e.g. first issue of document
1.1.1	12/03/2015	Alicja Kawecki, TM Forum	Updated cover, footer and Notice to reflect TM Forum Approved status
2.0.0	08-Nov-2017	Peter Norbury	Update to Notice and change document number
2.0.1	12-Dec-2017	Adrienne Walcott	Formatting/style edits prior to publishing
2.0.2	20-Mar-2018	Adrienne Walcott	Updated to reflect TM Forum Approved Status

Release History

This section records the changes between this and the previous Official document release. The release number is the 'Marketing' number which this version of the document is first being assigned to.

Release Number	Date Modified	Modified by:	Description of changes
17.5.0	08-Nov-2017	Peter Norbury	First release of document
17.5.1	20-Mar-2018	Adrienne Walcott	Updated to reflect TM Forum Approved Status

Acknowledgments

This document was prepared by the members of the TM Forum [team name] team:

- Pierre Gauthier, TM Forum, **Editor** and Team Leader
- Luis Velarde , Telefonica, Author