Frameworx Specification

REST API Design Guidelines
Part 2

Advanced guidelines for RESTful APIs lifecycle management, polymorphism, common tasks

TMF631
Release 14.5.1
March 2015

<table>
<thead>
<tr>
<th>Latest Update: Frameworx Release 14.5</th>
<th>TM Forum Approved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.1.1</td>
<td>IPR Mode: RAND</td>
</tr>
</tbody>
</table>

©TM Forum 2015. All Rights Reserved.
Notice

Copyright © TM Forum 2015. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to TM FORUM, except as needed for the purpose of developing any document or deliverable produced by a TM FORUM Collaboration Project Team (in which case the rules applicable to copyrights, as set forth in the TM FORUM IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by TM FORUM or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and TM FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

TM FORUM invites any TM FORUM Member or any other party that believes it has patent claims that would necessarily be infringed by implementations of this TM Forum Standards Final Deliverable, to notify the TM FORUM Team Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this deliverable.

The TM FORUM invites any party to contact the TM FORUM Team Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this TM FORUM Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this TM FORUM Standards Final Deliverable. TM FORUM may include such claims on its website, but disclaims any obligation to do so.

TM FORUM takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this TM FORUM Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on TM FORUM's procedures with respect to rights in any document or deliverable produced by a TM FORUM Collaboration Project Team can be found on the TM FORUM website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this TM FORUM Standards Final Deliverable, can be obtained from the TM FORUM Team.
Administrator. TM FORUM makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Direct inquiries to the TM Forum office:

240 Headquarters Plaza, 
East Tower – 10th Floor, 
Morristown, NJ 07960 USA 
Tel No. +1 973 944 5100 
Fax No. +1 973 944 5110 
TM Forum Web Page: [www.tmforum.org](http://www.tmforum.org)
Table of Contents

Notice.................................................................................................................................................. 2
Table of Contents .................................................................................................................................. 4
List of Figures ....................................................................................................................................... 5
List of Tables ......................................................................................................................................... 5
Executive Summary ............................................................................................................................ 6
1. Polymorphic Collections and Types ................................................................................................. 7
   Query using base collection .............................................................................................................. 8
2. Export and Import Data Tasks .......................................................................................................... 12
   Export Job Resource ....................................................................................................................... 12
   Import Job Resource ...................................................................................................................... 13
   Creating Export Jobs ..................................................................................................................... 15
   Creating Import Jobs ...................................................................................................................... 16
   Query Export Jobs ......................................................................................................................... 17
   Query Import Jobs ......................................................................................................................... 17
   Export Job Completion Notification ............................................................................................... 18
   Import Job Completion Notification ............................................................................................... 18
3. Depth Directive ................................................................................................................................. 20
4. Entity Versioning and Lifecycle Management .................................................................................... 24
   Query all versioned resources ......................................................................................................... 25
   Query a specific versioned resource ............................................................................................... 26
   Query current version of a resource ............................................................................................... 27
   Create new version of a resource ................................................................................................. 27
   Modify an existing version of a resource ....................................................................................... 28
   Role based Access Control ............................................................................................................ 30
5. Appendix A: Terms and Abbreviations Used within this Document ................................................ 31
   Terminology ................................................................................................................................... 31
   Abbreviations and Acronyms .......................................................................................................... 31
6. References ......................................................................................................................................... 32
   References ....................................................................................................................................... 32
7. Administrative Appendix ................................................................................................................... 33
   Document History ........................................................................................................................... 33
   7.0.1. Version History .................................................................................................................. 33
   7.0.2. Release History ................................................................................................................... 33
   Company Contact Details ............................................................................................................... 34
   Acknowledgments ............................................................................................................................ 34

© TM Forum 2015. All Rights Reserved.
List of Figures
N/A

List of Tables
N/A
Executive Summary

This document, “REST API Design Guidelines Part 2” provides information for the development of TM Forum APIs using REST.

It provides recommendations and guidelines for the implementation of the Polymorphic Operations, Common Export and Import Tasks and finally the Entity Lifecycle Management related patterns.
1. Polymorphic Collections and Types

In the following section we will use a simple example based on a Logical Resource Inventory Management system to illustrate the concept of polymorphic collections and types.

Anonymous collections are used to group resource of the same type or resources related to the same base type.

For example the party collection in a party management system can be used to group all the individual and organization resources.

- The name of a type in the design guideline represents the collection linked to the type of resource.
  - For example “tpe” represents the collection of all tpe resources while “link” represents the collection of all the “link” resources.
- TYPE is a reserved resource attribute like ID is.
  - For example a link entity will have a “type”=”link” attribute.
- The name of the TYPE attribute is the same as the name of the resource.
- Type scoping is implicit to the declaration of the collection type.
- The base type of an entity can be used to represent the collection of all entities with the same base type.
For example the “logicalResource” collection will scope both the “tpe” and “link” resources.

- The subtype can be used to represent the collection of all entities of a sub type.
  - For example “tpe” only represent the “tpe” resources.
- When querying on a base collection type only the most derived concrete resource representations are returned. The abstract resource is never returned.
  - GET /logicalResource will only return resource representation for “tpe or “link”. Never for “party”.
- A base collection can only be used as the base collection type to target entities related by the same super type.
- Every entity must be addressable via their proper collections. Only the most derived resource is the representation of the entity there are no other representations.
  - For example it may possible to do a GET /logicalResource/?type=”link”&”id”=”42” or GET /link/42 will return the same resources.
- Only concrete resources support the CUD uniform operations
  - POST operations are not supported against BASE TYPES
  - PUT operations are not supported against BASE TYPES
  - PATCH operations are not supported against BASE TYPES
  - DELETE operations are not supported against BASE TYPES
- Abstract resources may support GET operations
- Notifications are always relative to the concrete type or most derived resource representation.

Query using base collection

The following example operation shows how to retrieve collections of any logical resource items.
For example we can retrieve both tpe and links using this operation.

**REQUEST**

GET /api/inventory/logicalResource
Accept: application/json

**RESPONSE**

200

Content-Type: application/json

Content-Range: 1-3/3

[
  {
    "id": "1",
    "type": "tpe",
    "isBundle": false,
    "alias": [
      {
        "name": "CLI name",
        "value": "tpe22938"
      }
    ],
    "href": "http://server/api/inventory/tpe/1"
  },
  {
    "id": "2",
    "type": "tpe",
    "isBundle": false,
    "alias": [
      {
        "name": "CLI name",
        "value": "tpe22938"
      }
    ],
    "href": "http://server/api/inventory/tpe/2"
  },
  {
    "id": "3",
    "type": "link",
    "isBundle": false,
    "href": "http://server/api/inventory/link/3",
    "tp": [
      {
        "href": "http://server/api/inventory/tpe/1",
        "role": "AEND"
      }
    ]
  }
]
Or using ID based filtering to extract specific resources.

**REQUEST**

GET /api/inventory/logicalResource?id=1&id=3

Accept: application/json

**RESPONSE**

200

Content-Type: application/json

```
{
    "id": "1",
    "type": "tpe",
    "isBundle": false,
    "alias": [
        {
            "name": "CLI name",
            "value": "tpe22938"
        }
    ],
    "href": "http://server/api/inventory/tpe/1"
},
{
    "id": "3",
    "type": "link",
    "isBundle": false,
    "href": "http://server/api/inventory/link/3",
```
"tp": [
  {
    "href": "http://server/api/inventory/tpe/1",
    "role": "AEND"
  },
  {
    "href": "http://server/api/inventory/tpe/2",
    "role": "ZEND"
  }
]
2. Export and Import Data Tasks

This section describes the common tasks used for exporting and importing resources representations to files.

Two common task resources are defined:

- ImportJob used to import resources from a File
- ExportJob used to export resources to a File

Export Job Resource

An ExportJob resource represents a TASK used to export resources to a File.

The ExportJob resource supports the following properties:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Used to scope the exported data (identical to GET filter construct using target ID as base)</td>
</tr>
<tr>
<td></td>
<td>“query”: &quot;type=productOffering&amp;version=2.0”</td>
</tr>
<tr>
<td>path</td>
<td>URL of the root resource acting as the source for streaming content to the file specified by the ExportJob</td>
</tr>
<tr>
<td></td>
<td>../catalogManagement/catalog/42</td>
</tr>
<tr>
<td></td>
<td>i.e the relative path to the exportJob when it was created can also be used</td>
</tr>
<tr>
<td>content-type</td>
<td>The format of the exported data. By default “application/json”</td>
</tr>
<tr>
<td>status</td>
<td>notstarted, running, succeeded, failed</td>
</tr>
<tr>
<td>url</td>
<td>URL of the File containing the data to be exported. A file URL, which is of the form file://host/path where host is the fully qualified domain name of the system on which the path is accessible, and path is a hierarchical directory path of the form directory/directory/.../name</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>completionDate</td>
<td>Date at which the Job was completed.</td>
</tr>
<tr>
<td>creationDate</td>
<td>Date at which the Job was created.</td>
</tr>
<tr>
<td>errorLog</td>
<td>Reason for Failure</td>
</tr>
</tbody>
</table>

**Import Job Resource**

An ImportJob resource represent a TASK used to import resources from a File. The ImportJob resource supports the following properties:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content-type</td>
<td>The format of the imported data. By default \“application/json\”</td>
</tr>
<tr>
<td>path</td>
<td>URL of the root resource where the content of the</td>
</tr>
<tr>
<td><strong>file</strong> specified by the ImportJob must be applied</td>
<td>./catalogManagement/catalog/42</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>i.e the relative path to the importJob when it was created</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>status</strong></th>
<th>notstarted, running, succeeded, failed</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>url</strong></th>
<th>URL of the File containing the data to be imported</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a file URL, which is of the form file://host/path</td>
</tr>
<tr>
<td></td>
<td>where host is the fully qualified domain name of the system on which the path is accessible, and path is a hierarchical directory path of the form directory/directory/.../name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>completionDate</strong></th>
<th>Date at which the Job was completed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>creationDate</strong></td>
<td>Date at which the Job was created.</td>
</tr>
<tr>
<td><strong>errorLog</strong></td>
<td>Reason for Failure if status is failed</td>
</tr>
</tbody>
</table>
Creating Export Jobs

ExportJob Tasks are created as resources. The ExportJob is attached to a specific resource acting as the root for the collection of resources to be streamed to a File.

An ExportJob can be attached to a specific Catalog in a Catalog application or may be attached to the Product Offerings within a Catalog.

- ../catalogManagement/catalog/42/exportJob
  Export all the resources within the Catalog 42
- ../catalogManagement/catalog/42/productOffering/exportJob
  Export all the ProductOffering resources

For example:

REQUEST

POST catalogManagement/catalog/{10}/exportJob
Content-type: application/json

{
}

RESPONSE

201
Content-Type: application/json
Location: /api/catalogManagement/exportJob/54

{
  "id": "54",
  "href": "http://api/catalogManagement/exportJob/54",
  "status": "running",
  "path": "catalogManagement/catalog/10",
  "content-type": "application/json",
  "errorLog": "",
  "creationDate": "2013-04-19T16:42:23-04:00",
  "completionDate": "",
Creating Import Jobs

ImportJob Tasks are created as resources. The ImportJob maybe attached to the URL of the root resource where the content of the file specified by the ImportJob will be applied.

For example to apply the content of the import file to the catalog 10:

**REQUEST**

```
POST catalogManagement/catalog/{10}/importJob
Content-type: application/json

{}
```

**RESPONSE**

```
201
Content-Type: application/json
Location: /api/catalogManagement/importJob/554

{
  "id": "554",
  "href": "http://api/catalogManagement/importJob/554",
  "status": "running",
  "path": "catalogManagement/catalog/10",
  "content-type": "application/json",
  "errorLog": "",
  "creationDate": "2013-04-19T16:42:23-04:00",
  "completionDate": "",
}
```
**Query Export Jobs**

ExportJob resources can be found under the API/exportJob collection and may be retrieved using the normal GET constructs.

For example:

```
GET API/catalogManagement/exportJob/{10}
Accept: application/json
```

**RESPONSE**

```
{
    "id": "10",
    "status": "Succeeded"
}
```

---

**Query Import Jobs**

ImportJob resources can be found under the API/importJob collection and may be retrieved using the normal GET constructs.

```
GET /catalogManagement/importJob/{25}
Accept: application/json
```

**RESPONSE**

```
{
    "id": "25",
    "status": "Succeeded"
}
```
Export Job Completion Notification

This event provides notification that an export task has been completed.

For example:

REQUEST

```plaintext
POST /client/listener
Accept: application/json

{
    "event": {
        "id": "54",
        "href": "http://api/catalogManagement/exportJob/54",
        "status": "running",
        "path": "catalogManagement/catalog\{10\}",
        "contentType": "application/json",
        "errorLog": "",
        "creationDate": "2013-04-19T16:42:23-04:00",
        "completionDate": "",
    },
    "eventType": "exportJobCompleted"
}
```

Import Job Completion Notification

This event provides a notification that an import task has been completed.

REQUEST

```plaintext
POST /client/listener
Accept: application/json
```
3. Depth Directive

A common use case is to query a resource and “dereference” or “embed” the hyperlinked resource representations. This is achieved by issuing a GET with the DEPTH directive.

The GET /api/<resource>/?DEPTH=n operation is used to expand inline the referenced data up to the level of the specified depth level.

In the following example the local represented the dereferenced data for href: "http://server/api/shelf/7788"

```
GET /catalogManagement/productOffering/23/?depth=3
Accept: application/json
```

**RESPONSE**

```
Content-Range:items 23-24/50

[
  {
    "id": "23",
    "version": "2.0",
    "lastUpdate": "2013-04-19T16:42:23-04:00",
    "name": "Sensor mini",
    "description": "A wireless sensor for small garden",
    "isBundle": "false",
    "lifecycleStatus": "Active",
    "validFor": {
      "startDateTIme": "2013-04-19T16:42:23-04:00",
      "endDateTIme": "2013-06-19T00:00:00-04:00"
    }
  },
  {
    "id": "14",
    "href": "http://serverlocation:port/catalogManagement/category/14",
    "version": "2.0",
    "name": "Wireless sensors"
  }
]```

{  
  "channel": [  
    {  
      "id": "13",  
      "href":  
        "http://serverlocation:port/marketSales/channel/13",  
      "name": "IOT Corner"  
    }  
  ],  
  "place": [  
    {  
      "id": "12",  
      "href":  
        "http://serverlocation:port/marketSales/place/12",  
      "name": "France"  
    }  
  ],  
  "serviceLevelAgreement": {  
    "id": "28",  
    "href":  
    "name": "Standard SLA"  
  },  
  "productSpecification": [  
    {  
      "id": "13",  
      "href":  
      "version": "2.0",  
      "name": "wireless sensor mini",  
      "productSpecCharacteristic": [  
        {  
          "id": "34",  
          "name": "Colour",  
          "description": "Colour",  
          "valueType": "string",  
          "configurable": "true",  
          "validFor": {  
            "startDate": "2013-04-19T16:42:23-04:00",  
            "endDate": ""  
          },  
          "ProductSpecCharacteristicValue": [  
            
          ]  
        }  
      ]  
    }  
  ]
}
{ "valueType": "string", "default": "false", "value": "Black", "unitOfMeasure": "", "valueFrom": "", "valueTo": "", "validFor": { "startDateTime": "2013-04-19T16:42:23-04:00", "endDateTime": "" } }, { "valueType": "string", "default": "false", "value": "White", "unitOfMeasure": "", "valueFrom": "", "valueTo": "", "validFor": { "startDateTime": "2013-04-19T16:42:23-04:00", "endDateTime": "" } } ], "productOfferingPrice": [ { "id": "15", "href": "http://serverlocation:port/catalogManagement/productOfferingPrice/15", "name": "Sale Price", "description": "Sale price", "validFor": { "startDateTime": "2013-04-" } } } ]}
19T16:42:23-04:00", "endDateTime": "2013-06-19T00:00:00-04:00"
],

"priceType": "one time",
"unitOfMeasure": "",
"price": {
  "taxIncludedAmount": "12.00",
  "dutyFreeAmount": "10.00",
  "taxRate": "20.00",
  "currencyCode": "EUR"
},

"recurringChargePeriod": "",
"productOfferPriceAlteration": {
  "id": "15",
  "name": "Shipping Discount",
  "description": "One time shipping discount",
  "validFor": {
    "startDateTime": "2013-04-19T16:42:23-04:00",
    "endDateTime": ""
  },
  "priceType": "One Time discount",
  "unitOfMeasure": "",
  "price": {
    "percentage": "100%"
  },
  "recurringChargePeriod": "",
  "applicationDuration": "",
  "priceCondition": "apply if total amount of the order is greater than 300.00"
}
}
4. Entity Versioning and Lifecycle Management

In Product Lifecycle Management there is a requirement to distinguish between entities existing with different PLM version numbers and accessible via different ACL mechanisms.

For example the same Product Offerings may exist in a Catalog but with different version numbers.

It may be possible for an administrator to see all the existing versions or for a partner to see only a subset of all the existing versions.

The entity version number is not dependent on the version number of the API. For example in PLM the same API (running at a specific version number) may be used to retrieve entities with different PLM version numbers.

In order to distinguish resources representing entities running with different version numbers and accessible though the same API version the following directive can be used /id:(version=x) and the version attribute is added to each entity.

```
{
    "id": "42",
    "href": "http://serverlocation:port/catalogManagement/productOffering/42",
    "version": "1.0",
    "lastUpdate": "2013-04-19T16:23-04:00",
    "name": "Virtual Storage Medium",
    "description": "Virtual Storage Medium",
    "isBundle": "true",
    "lifecycleStatus": "Active",
    ...
}
```

Note that the resources in this case may have the same ID but may be distinguished by the inclusion of the version number in their ID i.e
In the following examples we will assume that two versions of the VirtualStorage Product Offer exist in the Product Catalog an Inactive and Active version respectively version 1.0 and version 2.0.

**Query all versioned resources**

Admin user of Catalog have access to all the versions of the resources while non admin users have by default access to only the latest version of the entities in the Catalog.

Users with different roles may have access to different versions of the entities in the catalog. For example user A may have access to only the version 1.0 of the entities while user B may have access to version 1.0 and version 2.0.

For example the following request on the admin endpoint will return all the versioned resources matching a specific ID.

<table>
<thead>
<tr>
<th>REQUEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET api/admin/catalogManagement/productOffering/?id=VirtualStorage</td>
</tr>
<tr>
<td>Accept: application/json</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
</tr>
<tr>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td>[{</td>
</tr>
</tbody>
</table>
| "id": "VirtualStorage",
| "href": "http://serverlocation:port/catalogManagement/productOffering/VirtualStorage",
| "version": "1.0",
| "lastUpdate": "2013-04-19T16:42:23-04:00",
| "name": "Virtual Storage Medium",
| "description": "Virtual Storage Medium",
| "isBundle": "true" |
"lifecycleStatus": "InActive",
,
},
{
"id": "VirtualStorage",
"href": "http://serverlocation:port/catalogManagement/productOffering/VirtualStorage",
"version": "2.0",
"lastUpdate": "2013-04-19T16:42:23-04:00",
"name": "Virtual Storage Medium",
"description": "Virtual Storage Medium",
"isBundle": "true",
"lifecycleStatus": "Active",
,
}
]

---

**Query a specific versioned resource**

In general a non admin API user only has visibility to the latest version number or visibility to a subset of versioned resources.

It may be possible for an admin API user to retrieve a resource with a specific version number by using an ID and versioning filtering criteria.

**REQUEST**

GET
api/admin/catalogManagement/productOffering/?id=VirtualStorage&version=1.0
Accept: application/json

**RESPONSE**

200
Content-Type: application/json

[{
"id": "42",
"href": "http://serverlocation:port/catalogManagement/productOffering/42",
"version": "1.0",
"lastUpdate": "2013-04-19T16:42:23-04:00",
"name": "Virtual Storage Medium",
"description": "Virtual Storage Medium",
"isBundle": "true",
"lifecycleStatus": "Active",

]

---

**Query current version of a resource**

By default only the most current version is returned (for admin and non admin).

<table>
<thead>
<tr>
<th>REQUEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET api/admin/catalogManagement/productOffering/VirtualStorage</td>
</tr>
<tr>
<td>Accept: application/json</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
</tr>
<tr>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td>{</td>
</tr>
</tbody>
</table>
|   "id": "42",
|   "href": "http://serverlocation:port/catalogManagement/productOffering/42",
|   "version": "2.0",
|   "lastUpdate": "2013-04-19T16:42:23-04:00",
|   "name": "Virtual Storage Medium",
|   "description": "Virtual Storage Medium",
|   "isBundle": "true",
|   "lifecycleStatus": "Active",
|   ...
| } |

---

**Create new version of a resource**

POST is used to create a new version of a resource.

The constraint is that the version numbers for the resource having the same ID must differ.
REQUEST
POST catalogManagement/productOffering
Content-type: application/json

{
    "id": "VirtualStorage",
    "version": "3.0",
    "name": "Virtual Storage Medium",
    "description": "Virtual Storage Medium",
    "isBundle": "true",
    "lifecycleStatus": "Active",
    "validFor": {
        "startDateTime": "2013-04-19T16:42:23-04:00",
        "endDateTime": "2013-06-19T00:00:00-04:00"
    },
    ...
}

RESPONSE
201
Content-Type: application/json

{
    "id": "VirtualStorage",
    "href": "http://serverlocation:port/catalogManagement/productOffering/42",
    "version": "3.0",
    "lastUpdate": "2013-04-19T16:42:23-04:00",
    "name": "Virtual Storage Medium",
    "description": "Virtual Storage Medium",
    "isBundle": "true",
    "lifecycleStatus": "Active",
    "validFor": {
        "startDateTime": "2013-04-19T16:42:23-04:00",
        "endDateTime": "2013-06-19T00:00:00-04:00"
    },
    ...
}

Modify an existing version of a resource

By default PATCH or PUT will be acting only on the latest version of the Resource. For example PATCH /.../productOffering/VirtualStorage will only update the VirtualStorage ProductOffering at Version 2.0 (which is the most current).
To update a specific version of an entity the (Version=X) directive is added to the ID (i.e. /id:(version=x)).

Note that this capability is only available to API users having the proper authorizations to change the catalog entities with specific version numbers.

For example to change the VirtualStorage versioned at 1.0 we could use 
/productOffering/VirtualStorage(Version=1.0)

**REQUEST**

PATCH /catalogManagement/productOffering/VirtualStorage(Version=1.0)
Content-type: application/json

```
{
    "lifecycleStatus": "Active"
}
```

**RESPONSE**

201
Content-Type: application/json

```
{
    "id": "VirtualStorage",
    "href":
    "http://serverlocation:port/catalogManagement/productOffering/VirtualStorage",
    "version": "1.0",
    "lifecycleStatus": "Active",

    ...
}
```
Role based Access Control

The user presents their credentials for authentication

If the credentials are valid

1. The user is given access to the catalog
2. As defined by their role(s)
3. As defined by their access rights
4. As defined by the access type: CRUD, discover
5. As defined by the pre-defined filter

For example if they issue a get on a catalog that a party has no access they get an error response

Or if they try to modify an area of the catalog but do not have Write Access they get an error response

We anticipate that OAUTH2 or Open ID Connect will be used as the authorization APIs and that ACL are established between authorized parties with regards to the content of the Catalog (i.e. GET but also enable of update operations on specific entities).
5. Appendix A: Terms and Abbreviations Used within this Document

**Terminology**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>TMF or Outside Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;BA Specific Term 1&gt;&gt;</td>
<td>&lt;&lt;Definition 1&gt;&gt;</td>
<td>&lt;&lt;Source&gt;&gt;</td>
</tr>
<tr>
<td>&lt;&lt;BA Specific Term n&gt;&gt;</td>
<td>&lt;&lt;Definition n&gt;&gt;</td>
<td>&lt;&lt;Source&gt;&gt;</td>
</tr>
</tbody>
</table>

**Abbreviations and Acronyms**

<table>
<thead>
<tr>
<th>Abbreviation/ Acronym</th>
<th>Abbreviation/ Acronym Spelled Out</th>
<th>Definition</th>
<th>TMF or External Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;Abbreviation/ Acronym 1&gt;&gt;</td>
<td>&lt;&lt;Expansion of abbreviation/acronym m 1&gt;&gt;</td>
<td>&lt;&lt;Definition 1&gt;&gt;</td>
<td>&lt;&lt;Source&gt;&gt;</td>
</tr>
<tr>
<td>&lt;&lt;Abbreviation/ Acronym n&gt;&gt;</td>
<td>&lt;&lt;Expansion of abbreviation/acronym m n&gt;&gt;</td>
<td>&lt;&lt;Definition n&gt;&gt;</td>
<td>&lt;&lt;Source&gt;&gt;</td>
</tr>
</tbody>
</table>
## 6. References

### References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
<th>Source</th>
<th>Brief Use Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Charter</td>
<td>&lt;&lt;PROJECT name&gt;&gt; Project Charter</td>
<td></td>
<td>&lt;&lt;summary&gt;&gt;</td>
</tr>
<tr>
<td>Link To Models</td>
<td>&lt;&lt; Hyperlinks to or location of associated models&gt;&gt;</td>
<td></td>
<td>&lt;&lt;summary&gt;&gt;</td>
</tr>
<tr>
<td>&lt;&lt;Reference 1&gt;&gt;</td>
<td>&lt;&lt;Type, Title, Number, Revision, Date&gt;&gt;</td>
<td></td>
<td>&lt;&lt;summary&gt;&gt;</td>
</tr>
<tr>
<td>&lt;&lt;Reference n&gt;&gt;</td>
<td>&lt;&lt; Type, Title, Number, Revision, Date&gt;&gt;</td>
<td></td>
<td>&lt;&lt;summary&gt;&gt;</td>
</tr>
</tbody>
</table>
7. 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**

7.0.1. 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.

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Date Modified</th>
<th>Modified by:</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>23/11/2014</td>
<td>Pierre Gauthier, TM Forum</td>
<td>Description e.g. first issue of document</td>
</tr>
<tr>
<td>1.1.1</td>
<td>12/03/2015</td>
<td>Alicja Kawecki, TM Forum</td>
<td>Updated cover, footer and Notice to reflect TM Forum Approved status</td>
</tr>
</tbody>
</table>

7.0.2. 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.

<table>
<thead>
<tr>
<th>Release Number</th>
<th>Date Modified</th>
<th>Modified by:</th>
<th>Description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>DD/MMM/YY</td>
<td>&lt;&lt;name&gt;&gt;</td>
<td>first release of document</td>
</tr>
</tbody>
</table>
Company Contact Details

<table>
<thead>
<tr>
<th>Company</th>
<th>Team Member Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include all involved companies adding lines as necessary.</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td>Phone</td>
</tr>
<tr>
<td></td>
<td>Fax</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Title</td>
</tr>
<tr>
<td></td>
<td>Email</td>
</tr>
<tr>
<td></td>
<td>Phone</td>
</tr>
<tr>
<td></td>
<td>Fax</td>
</tr>
</tbody>
</table>

Acknowledgments

This document was prepared by the members of the TM Forum <<team name>> team:

- Pierre Gauthier, TM Forum, Editor and Team Leader
- Maxime Delon Orange
- Veronique Mauneau Orange
- John Morey Ciena