TM Forum Specification

TMF630 REST API Design Guidelines Part 2

Advanced guidelines for RESTful APIs polymorphism, extension patterns, depth and expand directive, entity RefOrValue

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 2” provides information for the development of TM Forum APIs using REST.

It provides recommendations and guidelines for the implementation of the Polymorphic Operations, extension, depth and expand, EntityRefOrValue related patterns.

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].

Chapter 1. Polymorphic Collections and Types

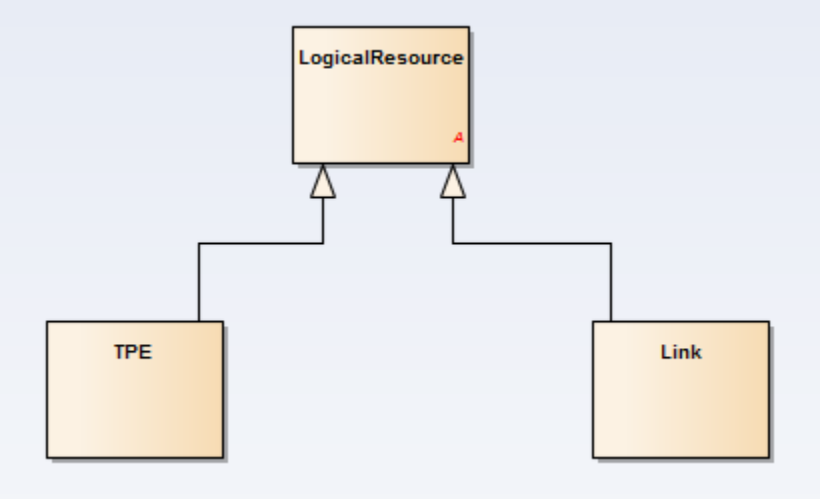
In the following section we will use a simple example based on a Logical Resource Inventory Management system to illustrate the concept of polymorphic collections and types.

The name of a type in the design guideline represents the collection linked to the type of resource.

“Tpe” represents the collection of all Tpe resources while “Link” represents the collection of all the “Link” resources.

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

For example, the logical resource collection in a resource management system can be used to group all the resources of type Tpe and Link.



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

- @type is a reserved resource attribute (ID is another reserved resource attribute) The value of the @type attribute is the same as the name of the resource.
  - For example, a link entity will have a “@type”=”Link” attribute.
- @type scoping is implicit to the declaration of the collection type.
- The @baseType 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, all concrete resource representations compatible with the base type are returned.
- Objects of abstract resources are never returned (since by definition an abstract resource cannot be instantiated).

`GET /logicalResource` will only return resource representations for “Tpe or “Link” if “LogicalResource” is an abstract type.

Example:

REQUEST

```
GET /resourceInventoryManagement/logicalResource?@type=Link&@type=Tpe
Accept: application/json
```

RESPONSE

```
HTTP 200
```

```
[
  {
    "id": "42",
    "href": "http://server:port/resourceInventoryManagement/logicalResource/42",
    "category": "",
    "@type": "Link",
    "@baseType": "LogicalResource",
    "@schemaLocation":
"//server:port/resourceInventoryManagement/schema/Link.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href":
"http://server:port/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "42",
        "href": "http://serverlocation:port/customerManagement/individual/42"
      }
```

```
    ]
  },
  {
    "id": "44",
    "href": "http://server:port/resourceInventoryManagement/logicalResource/44",
    "category": "",
    "@type": "Tpe",
    "@basetype": "LogicalResource",
    "@schemaLocation":
"//server:port/resourceInventoryManagement/schema/Tpe.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href":
"http://server:port/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "44",
        "href": "http://serverlocation:port/customerManagement/individual/44"
      }
    ]
  }
]
```

All operations SHOULD be relative to the base types.

Other collections MAY be used at the discretion of the API Designer. In this case the operations exposed on the base collection may or may not be present on the derived collection.

Example:

- `GET /resource/42?@type=Link`
- `GET/ logicalResource/42?@type=Link`
- `GET /link/42`

An entity MUST declare in the Header the alternative URI that can be used to identify it. E.g / link/42 for a logical resource of type link.

ID MUST always be relative to the base collection.

1.1. 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 /resourceInventoryManagement/logicalResource?@type=Link&@type=Tpe
Accept: application/json
```

RESPONSE

```
200
Content-Type: application/json
Content-Range: 1-2/2
```

```
[
  {
    "id": "42",
    "href": "http://server:port/resourceInventoryManagement/logicalResource/42",
    "category": "",
    "@type": "Link\"",
    "@basetype": "LogicalResource",
    "@schemaLocation":
"//server:port/resourceInventoryManagement/schema/Link.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href":
"http://server:port/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "42",
        "href": "http://serverlocation:port/customerManagement/individual/42"
      }
    ]
  },
  {
    "id": "44",
```

```
    "href": "http://server:port/resourceInventoryManagement/logicalResource/44",
    "category": "",
    "@type": "Tpe",
    "@basetype": "LogicalResource",
    "@schemaLocation":
"//server:port/resourceInventoryManagement/schema/Tpe.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href":
"http://server:port/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "44",
        "href": "http://serverlocation:port/customerManagement/individual/44"
      }
    ]
  }
]
```

Or using ID based filtering to extract specific resources.

REQUEST

```
GET /resourceInventoryManagement/logicalResource?id=42&id=44
Accept: application/json
```

RESPONSE

```
200
Content-Type: application/json
```

```
[
  {
    "id": "42",
    "href": "/resourceInventoryManagement/logicalResource/42",
    "category": "",
    "@type": "Link\"",
    "@basetype": "LogicalResource",
    "@schemaLocation": "/resourceInventoryManagement/schema/Link.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href": "/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "42",
        "href": "/customerManagement/individual/42"
      }
    ]
  },
  {
    "id": "44",
    "href": "/resourceInventoryManagement/logicalResource/44",
    "category": "",
    "@type": "Tpe",
    "@baseType": "LogicalResource",
    "@schemaLocation": "/resourceInventoryManagement/schema/Tpe.yml",
    "lifecyleState": "Active",
    "resourceSpecification": {
      "id": "4",
      "href": "/resourceCatalogManagement/resourceSpecification/4"
    },
    "relatedParty": [
      {
        "role": "owner",
        "id": "44",
        "href": "/customerManagement/individual/44"
      }
    ]
  }
]
```

1.2. CUD operations using base collection

The following example shows how a concrete resource can be created using the base collection.

For example we can create a resource of type MSISDN in the inventory.

REQUEST

```
POST /resourceInventoryManagement/logicalResource
Content-Type: application/json
```

```
{
  "value": "07865443255 ",
  "@type": "MSISDN"
}
```

RESPONSE

```
201
```

```
 { // JSON Resource Representation with every provided and default attribute }
```

The following example shows how a concrete resource can be updated using the base collection.

For example we can change the lifecycleState of a logical resource of type MSISDN in the inventory.

REQUEST

```
PATCH /resourceInventoryManagement/logicalResource/42
Content-Type: application/json-patch+json
```

```
{
  "path": "/lifecycleState",
  "value": "Reserved",
  "op": "replace"
}
```

RESPONSE

```
200
```

```
 { // Similar JSON as in GET response with state changed }
```

The following example shows how a concrete resource can be updated using the base collection.

For example, we can delete a logical resource with id 42.

REQUEST

```
DELETE /resourceInventoryManagement/logicalResource/42
```

RESPONSE

```
204
```

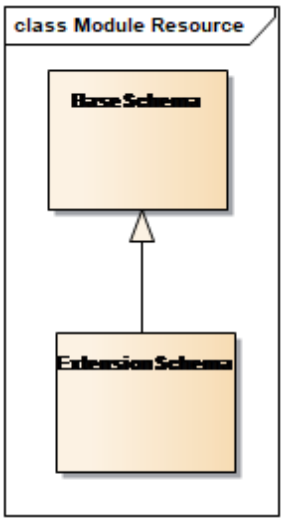
Chapter 2. Extension patterns

An extension is considered conformant or valid if it does not invalidate the core characteristics of an API and does not contradict the API Design Guidelines.

Various extension mechanisms are supported:

- Extending the basic schema of an entity like Product to create a subclass; in that case all the mandatory attributes and relationships of the base schema should be present in the extension.
- Adding characteristics to define a run time extension
- Adding relationships

It is not recommended to extend an TMF Open API with opaque name value pairs. Instead schema defined characteristic MUST be used.



2.1. Schema based extension

In the following section we will use a simple example to illustrate the concept of adding simple or complex characteristics to define run time extensions.

@schemaLocation is a reserved attribute (like @type and id),it provides a link to the schema which describes a REST resource.

An extension schema MUST include all the attributes of the base schema.

The following example shows how a physical resource can be extended with new attributes (operatingState attribute) at run-time:

```
{
    "id": "45",
    "href": "/resourceInventoryManagement/physicalResource/45",
    "publicIdentifier": "07467223333",
    "@type": "Equipment",
    "@baseType": "PhysicalResource",
    "@schemaLocation": "/resourceInventoryManagement/schema/Equipment.yml",
    "category": "Category 1",
    "lifecyleState": "Active",
    "manufactureDate": "2007-04-12",
    "serialNumber": "123456745644",
    "versionNumber": "11",
    "operatingState": "Working",
    "resourceSpecification": {
        "id": "6",
        "href": "/resourceCatalogManagement/resourceSpecification/6",
        "@type": "PhysicalResourceSpecification"
    },
    "relatedParty": [{
        "role": "Manufacturer",
        "id": "43",
        "href": "/PartyManagement/individual/43"
    }],
    "resourceAttachment": [{
        "href": "/documentManagement/document/123"
    }],
    "note": [{
        "text": "something about this resource"
    }],
    "place": {
        "id": "1979",
        "href": "/genericCommon/place/1979",
        "name": "Main Office",
        "role": "default delivery"
    }
}
```

All resources defined within an API MAY be extended, therefore attributes @type, @baseType and @schemaLocation MUST be supported optionally in the structure definition of all resources.

Those resources that are not expected to be extended are not required to explicitly include the attributes as part of the resource structure definition in the API spec documentation.

As an example, the following resources will be considered as expected to be extended

- Party: In SID they are defined as either Individual or Organization.
- PartyRole: In SID they are specialized to as Customer, Partner, Employee, etc.
- GeographicAddress: In SID it can specialized to AustralianPropertyAddress, JapanesePropertyAddress, AustralianRuralAddres, etc.
- Characteristic: The structure of the characteristics object may depend on the implementation or the element whose characteristics are to be described.

As part of the API Data model the resources that have been identified as expected to be extended in implementations must include parameters @type, @baseType and @schemaLocation explicitly within the structure defined for those resources.

In Swagger 2.0 and 3.0 `allOf` SHOULD be used to extend an object.

Here is an example of Equipment.yml schema for the example above:

```
PhysicalResource:
  title: PhysicalResource
  type: object
  properties:
    id:
      type: integer

Equipment:
  title: Equipment
  type: object
  allOf:
    - $ref: "#/definitions/PhysicalResource"
  - properties:
    operatingState:
      type: string
```

2.2. Schema based extension for the Characteristic of an entity

Assuming that Characteristic/CharacteristicSpec pattern is used, the following example shows how we can extend an API at run time by adding simple or complex characteristics:

```
{
  "id": "45",
  "href": "/resourceInventoryManagement/physicalResource/45",
  "publicIdentifier": "07467223333",
  "@type": "Equipment",
  "@baseType": "PhysicalResource",
```

```
  "@schemaLocation":
"http://server:port/resourceInventoryManagement/schema/Equipment.yml",
  "category": "Category 1",
  "lifecyleState": "Active",
  "manufactureDate": "2007-04-12",
  "serialNumber": "123456745644",
  "versionNumber": "11",
  "operatingState": "Working",
  "resourceSpecification": {
    "id": "6",
    "href": "/resourceCatalogManagement/resourceSpecification/6",
    "@type": "PhysicalResourceSpecification"
  },
  "resourceCharacteristic": [{
    "name": "physicalPort",
    "valueType":"Object",
    "value": {
      "@type": "PhysicalPort",
      "@schemaLocation": "http://host:port/schema/PhysicalPort.yml",
      "name": "LAN Port",
      "isActive": true
    }
  },
    {
      "name": "color",
      "value": "red"
    }],
  "resourceRelationship": [{
    "type": "requires",
    "resource": {
      "id": "46",
      "href": "/resourceInventoryManagement/logicalResource/46"
    },
    "resourceRelationshipCharacteristic": [{
      "name": "priority",
      "value": 2
    },
      {
        "name": "accuracy",
        "valueType":"Object",
        "value": {
          "@type": "Accuracy",

"@schemaLocation":"http://server:port/resourceInventoryManagement/schema/Accuracy
.yml",
```

```
          "unit": "second",
          "amount": "5"
        }
      }]
  }],
  "relatedParty": [{
    "role": "Manufacturer",
    "id": "43",
    "href": "/PartyManagement/individual/43"
  }],
  "resourceAttachment": [{
    "id": "/documentManagement/document/123"
  }],
  "note": [{
    "text": "something about this resource"
  }],
  "place": {
    "id": "1979",
    "href": "/genericCommon/place/1979",
    "name": "Main Office",
    "role": "default delivery"
  }
}
```

In the above example, ResourceCharacteristics class is a list of simple and complex characteristics where:

name	A string. Name of the characteristic.
value	A string (StringorObject). Value of the characteristic which can be simple or complex (object)
@schemaLocation	A string. A link to the schema describing this characteristic
@type	A string. (Class) type of the characteristic
valueType	A string. A kind of value that the characteristic can take on, such as numeric, text and so forth.

For complex characteristics, the name of the characteristic is the name of the object while the value is a complex object and the schema describing the object is provided as a link in the @schemaLocation attribute.

The following example shows how a ResourceFunctionSpec can be extended using the same mechanism as above:

```
{
  "id": "43",
  "href": "/catalogManagement/ResourceFunctionSpec/43",
  "name": "Firewall",
  "description": "This resource specification ...",
  "@type": "ResourceFunctionSpec",

"@schemaLocation":"https://host:port/catalogManagement/schema/ResourceFunctionSpe
c.yml",
  "@baseType": "LogicalResourceSpec",
  "version": "3.2",
  "validFor": {
    "startDateTime": "2017-08-06T00:00",
    "endDateTime": "2018-03-07T00:00"
  },
  "lastUpdate": "2017-08-09T00:00",
  "lifecycleStatus": "Active",
  "isBundle": false,
  "category": "Security",
  "targetResourceSchema": {
    "@type": "ResourceFunction",

"@schemaLocation":"https://host:port/catalogManagement/schema/ResourceFunction.ym
l "
  },
  "attachment": [{
    "description": "This attachment ...",
    "href": "/documentManagement/attachment/22",
    "id": "22",
    "type": "Document",
    "url": "http://xxxxx"
  }],
  "relatedParty": [{
    "id": "8406",
    "href": "/partyManagement/organization/8406",
    "role": "Supplier",
    "name": "Firewall Express",
    "validFor": {
      "startDateTime": "2017-08-05T00:00",
      "endDateTime": "2018-03-07T00:00"
    }
  }],
  "resourceSpecCharacteristic": [{
    "name": "OperatingSystem",
```

```
    "description": "This resource spec characteristic ...",
    "valueType": "String",
    "@valueSchemaLocation": "",
    "configurable": true,
    "validFor": {
      "startDateTime": "2017-08-12T00:00",
      "endDateTime": "2018-03-07T00:00"
    },
    "@type": "ResourceSpecCharacteristic",

"@schemaLocation":"https://host:port/catalogManagement/schema/ResourceSpecCharact
eristic.yml",
    "minCardinality": 0,
    "maxCardinality": 1,
    "isUnique": true,
    "regex": "",
    "extensible": false,
    "resourceSpecCharRelationship": [{
      "type": "string",
      "name": "OperatingSystem",
      "id": "4690",
      "href": "/catalogManagement/resourceSpecification/4690",
      "@type": "ResourceFunctionSpec",
      "validFor": {
        "startDateTime": "2017-08-11T00:00",
        "endDateTime": "2018-03-07T00:00"
      }
    }],
    "resourceSpecCharacteristicValue": [{
      "isDefault": true,
      "value": "Android KitKat",
      "validFor": {
        "startDateTime": "2017-08-06T00:00",
        "endDateTime": "2018-03-07T00:00"
      }
    }]
  },
    {
      "name": "ScalabilityChar",
      "description": "Scalability parameters for this resource function",
      "valueType": "CapabilityScalable",
      "@valueSchemaLocation":
"https://host:port/catalogManagement/schema/CapabilityScalable.yml",
      "configurable": true,
      "validFor": {
```

```
        "startDateTime": "2017-08-17T00:00",
        "endDateTime": "2018-03-12T00:00"
      },
      "@type": "ResourceSpecCharacteristic",

"@schemaLocation":"https://host:port/catalogManagement/schema/ResourceSpecCharact
eristic.yml",
      "minCardinality": 0,
      "maxCardinality": 1,
      "isUnique": true,
      "extensible": true,
      "resourceSpecCharRelationship": [],
      "resourceSpecCharacteristicValue": [{
        "valueType": "Object",
        "value": {
          "minInstances": 1,
          "maxInstances": 1000
        },
        "isDefault": true,
        "validFor": {
          "startDateTime": "2017-08-17T00:00",
          "endDateTime": "2018-03-12T00:00"
        },
        "@type": "CapabilityScalable",
        "@schemaLocation":
"https://host:port/catalogManagement/schema/CapabilityScalable.yml"
      }]
    }],
  "resourceSpecRelationship": [{
    "type": "AdjacencyPair",
    "id": "2053",
    "href": "/catalogManagement/physicalBlackBoxSpec/2053",
    "name": "SWI.1",
    "validFor": {
      "startDateTime": "2017-08-08T00:00",
      "endDateTime": "2018-03-07T00:00"
    }
  }],
  "feature": [{
    "id": "8782",
    "name": "IPv4Addressing",
    "isBundle": false,
    "validFor": {
      "startDateTime": "2017-08-16T00:00",
      "endDateTime": "2018-03-12T00:00"
```

```
    },
    "isEnabled": true
  },
    {
      "id": "8783",
      "name": "IPv6Addressing",
      "isBundle": false,
      "validFor": {
        "startDateTime": "2017-08-16T00:00",
        "endDateTime": "2018-03-12T00:00"
      },
      "isEnabled": false
    }]
}
```

2.3. State extension pattern

Many instances represented in TMForum Open-APIs have a long-lived "state" and hence a "lifecycle". The collection of lifecycle states are typically described in an enumerated type that is applied to the **status** attribute. The rules of use and extension are as follows:

- The enumeration of states are considered "normative" but not comprehensive. They must not be removed from the specification, or semantically redefined by the implementation, but they can be added to - both in breadth (additional states) and depth (additional sub-states).
- The transitions between states are "illustrative". While the state-transition diagram provides a helpful graph, a compliant implementation is free to transition between states as it wishes - though obviously it would be helpful to document any variation for potential consumers.
- There is a distinction between the **operational state** of the Resource (such as *up*, *down*, *error*) and its **administrative state** (*planned*, *in-operation*, *retired*). These should not be confused or merged as they are independent of each other.

As a simple example, a ServiceProblem be *acknowledged,_inProgress* or *resolved* - so it's **status** attribute might be described as:

```
{
  "type": "string",
  "enum": [
    "acknowledged",
    "inProgress",
    "resolved"
  ]
}
```

Extending with new states:
Implementations might consider adding new states by extending the enumeration accordingly:

```
{
  "type": "string",
  "enum": [
    "acknowledged",
    "rejected",
    "inProgress",
    "resolved",
    "closed"
  ]
}
```

Extending with sub-states:
Instances of the resource should be expected to be in one of the "macro" states, but might chose to extend them to add more detail. Although a ServiceProblem is "**inProgress**" it might be waiting on an external action or information. A dotted notation can be used to describe appropriate sub-states such as **inProgress.held** or **inProgress.pending**.

```
{
  "type": "string",
  "enum": [
    "acknowledged",
    "rejected",
    "inProgress.held",
    "inProgress.pending",
    "resolved",
    "closed"
  ]
}
```

As a result, consumers should always consider the possible presence of sub-states, and search for (say) **inProgress(\..)?$** rather than "**inProgress**" (such as in JSON-Path: @.status == "**inProgress(\..)?$**"), which would match "**inProgress**", "**inProgress.pending**" (with **.pending** as a regex-group) but not "**inProgressPending**".

Chapter 3. Depth and Expand 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 and or an expand paramter.

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

The “depth” is used to expand the level of referenced (related) entities such as bundled offerings and “target” references of relations.

If depth=0, then all referenced entities just RootEntity objects which contains only ID and @type (may contain name and href).

If depth = 1, then the first level of related target entities is expanded. References are replaced with the actual loaded entities with the corresponding level of details.

If depth = N it expands reference objects of related entities recursively. The last level contains only the references.

NOTE	expanding should not lead to infinite recursion if there is a circular reference. In such case the object is expanded once when first met and at other places remains a reference.

“expand” parameter is used to narrow down (filter) expanding of the referenced objects.

If expand is not specified, then by default all the paths should be expanded. If the expand parameter is specified, only those paths should be expanded according to depth parameter.

If depth is not specified, then the default value will depend on the expand parameter. If expand parameter is set, then depth is 1. If not set, then 0.

In the following example the productOffering.productSpeficiation is expanded up to the level of depth specified by the depth parameter:

REQUEST

```
GET
/catalogManagement/productOffering/23/?depth=3&expand=productOffering.productSpec
ification
Accept: application/json
```

RESPONSE

```
{
  "id": "23",
  "href": "/catalogManagement/productOffering/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"
  },
  "category": [
    {
      "id": "14",
      "href": "/catalogManagement/category/14",
      "version": "2.0",
      "name": "Wireless sensors"
    }
  ],
  "channel": [
    {
      "id": "13",
      "href": "/marketSales/channel/13",
      "name": "IOT Corner"
    }
  ],
  "place": [
    {
      "id": "12",
      "href": "/marketSales/place/12",
      "name": "France"
    }
  ],
  "serviceLevelAgreement": {
    "id": "28",
    "href": "/slaManagement/serviceLevelAgreement/28",
    "name": "Standard SLA"
  },
  "productSpecification": [
    {
      "id": "13",
      "href": "/catalogManagement/productSpecification/13",
      "version": "2.0",
```

```
      "name": "wireless sensor mini",
      "productSpecCharacteristic": [
        {
          "id": "34",
          "name": "Colour",
          "description": "Colour",
          "valueType": "string",
          "configurable": "true",
          "validFor": {
            "startDateTime": "2013-04-19T16:42:23-04:00",
            "endDateTime": ""
          },
          "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": " /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",
        "href": " /catalogManagement/productOfferPriceAlteration/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"
      }
    }
  ]
}
```

Chapter 4. EntityRefOrValue pattern

When an object (or list of objects) exists in the server side then it can be referred to by its reference; when the object (or list of objects) does not exist then it can only be referenced by its value. The EntityRefOrValue pattern is useful when there is a need to pass to the server an object (or a list of objects) in either existence state.

For EntityRefOrValue all attributes are optional.

If it is a Ref, id and href are mandatory but if it is Value then only the mandatory parameters from the Entity referenced are required (id and href can be also added if the reference is used to modify the contents of an existing entity).

To apply this pattern to a plain entity (for instance Product), the corresponding ref entity needs to be defined in the API data model (in our example ProductRef).

The EntityRef MUST include necessarily an “id” and an “href” attributes, but other attributes from the plain entity may be included.

In the API data model

- Any class may be used as a RefOrValue. Therefore we should use the plain class in the API data model. E.g if we need to support RefOrValue pattern for Product class, we should link in the model to the Product class. A constraint on the relationship (refOrValue) indicates whether the pattern is actually used or not used.

4.1. Example:

In the following example the described pattern can be seen in the relation between entities Product and Place, Product and ProductRelationship and OrderItem and Product.

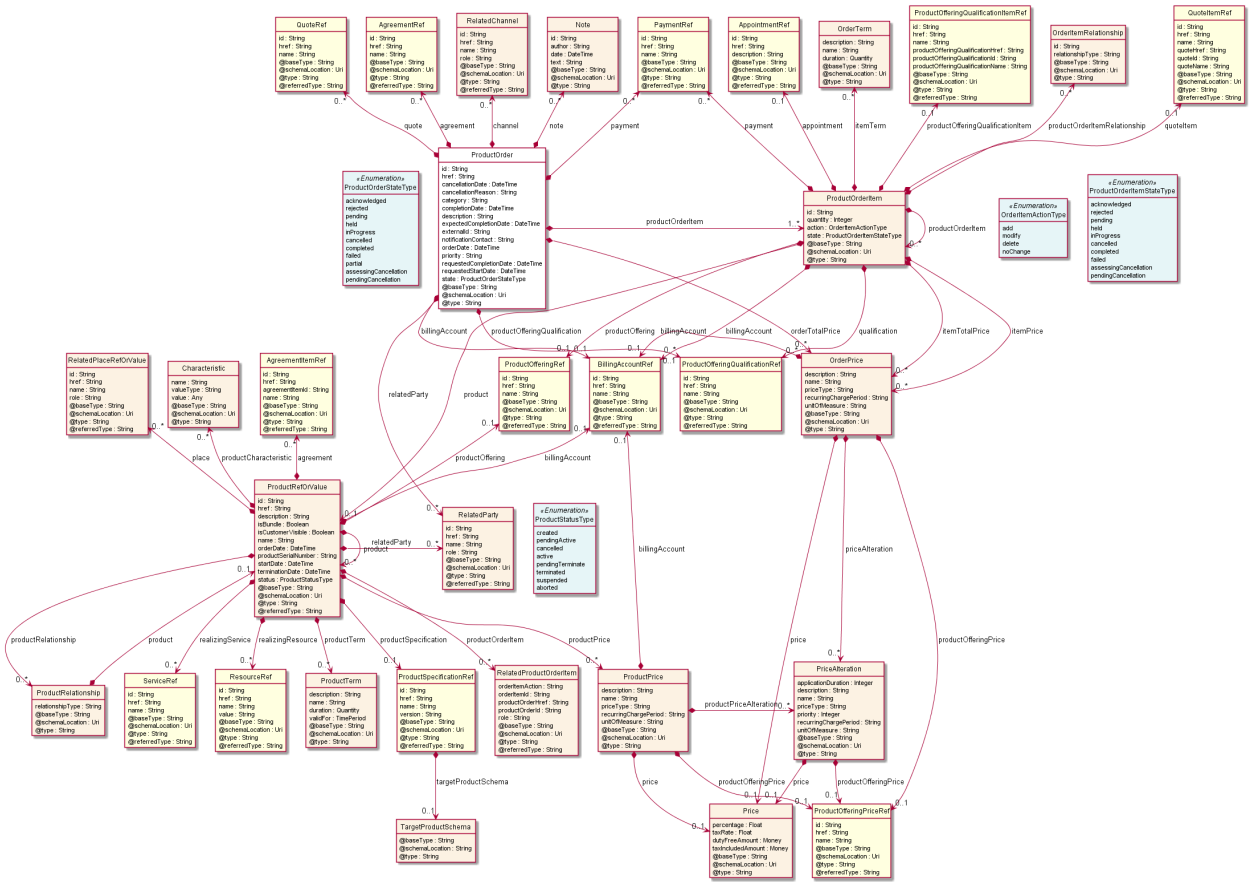


Figure . RefOrValue annotation used in API Data model

In Swagger implementation :

- Swagger 2.0 we should allow for a less strongly type solution. We will create the swagger schema as if it was a value and it will be up to the implementation to implement ref or value. All the parameters in the value class should be optional.

Example: In the following example the product class is defined as if it was a value.

```
...

"definitions": {
    ...
    "Product": {
        "description": "Configure the product characteristics (only configurable
            characteristics and necessary only if a non default value is
selected)
            and/or identify the product that needs to be modified/deleted.",
        "type": "object",
        "properties": {
```

```
            "id": {
                "description": "",
                "type": "string"
            },
            "href": {
                "description": "",
                "type": "string"
            },
            "name": {
                "description": "",
                "type": "string"
            },
            "place": {
                "description": "",
                "type": "array",
                "items": {
                    "$ref": "#/definitions/Place"
                }
            },
            "characteristic": {
                "description": "",
                "type": "array",
                "items": {
                    "$ref": "#/definitions/ProductCharacteristic"
                }
            },
            "relatedParty": {
                "description": "",
                "type": "array",
                "items": {
                    "$ref": "#/definitions/RelatedPartyRef"
                }
            },
            ...
            "productRelationship": {
                "description": "",
                "type": "array",
                "items": {
                    "$ref": "#/definitions/ProductRelationship"
                }
            },
            "productSpecification": {
                "description": "",
                "$ref": "#/definitions/ProductSpecificationRef"
            }
```

```
        }
    }
}
```

- In Swagger 3.0 we SHOULD make use of `oneOf` between value class and ref class.

Example: In the following example the Product is passed using oneOf.

```
{
  "ProductRefOrValue": {
    "oneOf": [
      {
        "$ref": "#/components/schemas/Product"
      },
      {
        "$ref": "#/components/schemas/ProductRef"
      }
    ]
  }
}
```

In this example we provide another example of how RefOrValue pattern is defined in Swagger 3.0 for relatedPartyRefOrValue:

```
{
  "relatedPartyRefOrValue": {
    "oneOf": [
      {
        "$ref": "#/components/schemas/Party"
      },
      {
        "$ref": "#/components/schemas/PartyRef"
      },
      {
        "$ref": "#/components/schemas/PartyRole"
      },
      {
        "$ref": "#/components/schemas/PartyRoleRef"
      }
    ]
  }
}
```

Here is a JSON example of ProductRefOrValue usage in Shopping Cart API where product is passed by value:

REQUEST

```
GET /shoppingCart/v1/shoppingCart/5074 +
Accept: application/json
```

RESPONSE

```
200
```


```
{
  "href": "/shoppingCart/v1/shoppingCart/1203",
  "id": "1203",
  "validFor": {
    "startDateTime": "2017-03-27T00:00",
    "endDateTime": "2017-10-24T00:00"
  },
  "contactMedium": [
    {
      "type": "email",
      "characteristic": {
        "email": "JackSmith@mail.com"
      }
    }
  ],
  "cartTotalPrice": [
    {
      "description": "Total Recurring Price.",
      "name": "Monthly Price",
      "priceType": "recurring",
      "recurringChargePeriod": "monthly",
      "price": {
        "dutyFreeAmount": {
          "money": {
            "unit": "EUR",
            "value": 29
          }
        },
        "taxIncludedAmount": {
          "money": {
```

```
            "unit": "EUR",
            "value": 31.9
          },
          "taxRate": 10
        }
      },
      "priceAlteration": [
        {
          "applicationDuration": 3,
          "description": "First 3 month get 50% off",
          "name": "3 month for half",
          "priceCondition": "new customer",
          "priceType": "recurring",
          "recurringChargePeriod": "monthly",
          "validFor": {
            "startDateTime": "2017-03-26T00:00",
            "endDateTime": "2017-10-24T00:00"
          },
          "price": {
            "percentage": 50
          }
        }
      ]
    }
  ],
  "cartItem": [
    {
      "action": "add",
      "id": "8307",
      "quantity": 1,
      "status": "Active",
      "note": {
        "author": "Mr Smith",
        "date": "2017-03-28T00:00",
        "text": "Please wrap with double bag"
      },
      "product": {
        "name": "Talk Simple 25",
        "place": [
          {
            "href": "/placeManagement/place/7396",
            "id": "7396",
            "name": "Main Store",
            "role": "default delivery"
          }
```

```
        ]
      },
      "productOffering": {
        "href": "/productCatalog/productOffering/14277",
        "id": "14277",
        "name": "Talk Simple 25"
      }
    }
  ]
}
```

Chapter 5. EntityRef pattern

This pattern is useful when there is a need to pass to the server an object or a list of objects by reference when the object(s) already exists on server side.

The EntityRef MUST include necessarily an “id” and an “href” attributes, but other attributes from the plain entity MAY be included like name, @type,@schemaLocation,validFor, role.

Use @type if required. If present, @type indicates the presence of a generalization (sub classing of Entity as per generic extension patterns).

The presence of @type is generally not needed for EntityRef objects.

Here is an example of where EntityRef pattern has been used for relatedPartyRef, ChannelRef, BillingAccountRef, PaymentRef.

```
{
  "id":"11",
  "href":"/productOrdering/productOrder/11.org",
  "externalId":"Telco01",
  "priority":"1",
  "description":"Order illustration UC1",
  "category":"Residential",
  "state":"Completed",
  "orderDate":"2017-11-03T08:46:47.945Z",
  "completionDate":"2017-11-03T08:46:47.945Z",
  "requestedStartDate":"2017-11-03T08:46:47.945Z",
  "requestedCompletionDate":"2017-11-03T08:46:47.945Z",
  "expectedCompletionDate":"2017-11-03T08:46:47.945Z",
  "notificationContact":"jean.pontus@tmforum.com",
  "@baseType":"ProductOrder",
  "@type":"ProductOrder",
  "ChannelRef":{
    "id":"1",
    "name":"store",
    "@referredType":"Store"
  },
  "note": {
    "text":"Order to illustrate UC1",
    "date":"2017-11-03T08:46:47.945Z",
    "author":"Jean-Lyuc Tymen"
  },
  "relatedParty": [{
    "id":"11",
    "href":"/tmfAPI/customer/11.org",
```

```
    "name":"Joe Doe",
    "role":"customer",
    "@referredType":"Customer"
  },
    {
      "id":"11",
      "href":"/tmfAPI/partyManagement/individual/11.org",
      "name":"Joe Doe",
      "@referredType":"Individual"
    },
    {
      "id":"12",
      "href":"/tmfAPI/customer/12.org",
      "name":"Arthur Pence",
      "role":"Seller",
      "@referredType":"Customer"
    }],
  "billingAccount":{
    "id":"BA1513",
    "href":"/tmfAPI/billingAccount/BA1513.org",
    "name":"BA01"
  },
  "orderItem":[{
    "id":"100",
    "action":"add",
    "state":"Completed",
    "quantity":1,
    "productOffering":{
      "id":"14277",
      "name":"TMF25"
    },
    "orderItem":[{
      "id":"110",
      "action":"add",
      "state":"Completed",
      "quantity":1,
      "itemPrice":[{
        "name":"AccessFee",
        "description":"Access Fee",
        "priceType":"oneTime",
        "price":{
          "taxIncludedAmount":{
            "value":0.99,
            "unit":"Euros"
          }
```

```
        }
      }],
      "productOffering": {
        "id":"14305",
        "name":"TMF Mobile Telephony"
      },
      "product": {
        "characteristic":[{
          "name":"TEL_MSISDN",
          "value":"27457958"
        }],
        "relatedParty":[{
          "id":"11",
          "href":"/tmfAPI/customer/11.org",
          "name":"Joe Doe",
          "role":"user",
          "@referredType":"Customer"
        }],
        "productSpecification": {
          "id":"14307",
          "href":"/tmfAPI/productSpecification/14307.org",
          "version":"1",
          "name":"Mobile Telephony"
        }
      },
      "payment":[{
        "id":"2365",
        "name":"shop payment cash #2365"
      }]
    },
      {
        "id":"120",
        "action":"add",
        "state":"Completed",
        "quantity":1,
        "itemPrice": [{
          "name":"monthly price",
          "description":"TMF Tariff Plan monthly fee",
          "priceType":"recurring",
          "recurringChargePeriod":"monthly",
          "price": {
            "taxIncludedAmount": {
              "value": 20,
              "unit":"Euros"
            },
```

```
            "taxRate":18.6
          },
          "priceAlteration":[{
            "name":"20Discount3Months",
            "description":"discount 20% for 3 months",
            "priceType":"recurring",
            "recurringChargePeriod":"monthly",
            "applicationDuration":"3",
            "priority":1,
            "price": {
              "percentage":20
            }
          }]
        }],
        "productOffering":{
          "id":"14344",
          "name":"TMF TariffPlan"
        },
        "product": {
          "productSpecification":{
            "id":"14395",
            "name":"TMF TariffPlan"
          }
        },
        "orderItemRelationship":[{
          "id":"110",
          "type":"reliesOn"
        }],
        "itemTerm": [{
          "name":"12MonthsCommitment",
          "description":"12 Months commitment",
          "duration": {
            "value":12,
            "unit":"string"
          }
        }]
      },
      {
        "id":"130",
        "action":"add",
        "state":"Completed",
        "quantity":1,
        "productOffering": {
          "id":"14354",
          "name":"CoverageOptions"
```

```
        },
        "product":{
          "characteristic":[{
            "name":"CoverageOption",
            "value":"National"
          }],
          "productSpecification": {
            "id":"14353",
            "name":"coverage"
          }
        },
        "orderItemRelationship":[{
          "id":"110",
          "type":"reliesOn"
        }]
      }]
  }]
}
```

Chapter 6. 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.

6.1. 6.1 Document History

6.1.1. 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.

Version	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-Mar-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

6.1.2. 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.

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

6.2. Acknowledgments

This document was prepared by the members of the TM Forum API team:

- Pierre Gauthier, TM Forum, **Editor** and Team Leader
- Nicoleta Stoica, Vodafone, Author
- Luis Velarde, Telefonica, Contributor
- Kamal Maghsoudlou, Ericsson, Contributor
- Mariano Belaunde, Orange, Contributor
- Ludovic Robert, Orange, Contributor